Fundamentals of ActionScript 3.0
DEVELOP AND DESIGN

Doug Winnie
This book is dedicated to Hoover. Hoover was a big part of my life, and was always by my side while doing “tech-no” things. I miss you Hoover!

This book is also dedicated to my husband, Mike. While not always into my “tech-no” things, he is my inspiration for doing great things—“tech-no” or not. Thanks, Groovy Dude!
With too many people to mention individually, I'm going to do a group acknowledgment of all of the members of the Adobe Flash Professional, Flash Player, AIR, and Flex teams that have inspired me to create great things, and to a platform that overcomes amazing obstacles to give creative and web professionals the ability to express themselves wherever they go.

I also want to acknowledge Chris Coudron, my tech reviewer and friend, for his effort to make me look good in code and for reminding me that there is not a NOR operator in ActionScript. D’oh!
## CONTENTS

Introduction .......................................................... xiii  
Welcome to ActionScript 3.0 ........................................ xvi

### PART 1 GETTING THE FUNDAMENTALS

#### CHAPTER 1  ACCESSING AND MANIPULATING OBJECTS 2
- Creating a New Project for ActionScript 3.0 .................. 4  
- Accessing Objects on the Stage ................................. 5  
- Sending Messages to the Output Panel ......................... 8  
- Working with Object Parameters ............................... 12  
- Wrapping Up ..................................................... 19

#### CHAPTER 2  DYNAMICALLY ADDING OBJECTS TO THE STAGE 20
- Creating Named Library Assets ................................ 22  
- Introducing the Display Stack .................................. 25  
- Creating New Objects on the Stage ......................... 25  
- Messing with the Display Stack ................................. 29  
- Assigning Properties to Dynamically Created Instances .. 30  
- Working with Comments ..................................... 31  
- Creating Comments ......................................... 31  
- Working with Spaces ....................................... 32  
- Wrapping Up ..................................................... 33

#### CHAPTER 3  WORKING WITH FUNCTIONS 34
- Fundamentals of Functions .................................. 36  
- Accepting Values in Functions ............................... 39  
- Possible Errors when Working with Functions .............. 42  
- Required Parameter Error .................................. 42  
- Type Mismatch Error ......................................... 43  
- Returning Values from Functions ......................... 44  
- More about Parentheses (and Curly Braces) ............... 46  
- Wrapping Up ..................................................... 47

#### CHAPTER 4  ACTIONSCRIPT AND MATH 48
- Mathematical Operators ...................................... 50  
- Addition and Subtraction .................................. 50
Using the Enter Frame Event ........................................ 104
Removing Event Listeners ........................................... 106
When to Use Frame vs. Timer Events ............................... 108
Wrapping Up ......................................................... 111

**PROJECT 1**

**COUNTDOWN CLOCK** ......................................... 112
Project Specification: Countdown Clock ......................... 114
Visual Design Review ................................................. 115
Kick-Off Meeting Notes: Countdown Clock .................... 116
Solution and Walkthrough: Countdown Clock .................. 117
*Project Setup in Flash Professional* .............................. 117
*ActionScript Setup* ................................................ 119
*Auto-Generated Imports* .......................................... 121
*Display Objects* ...................................................... 122
*Event Listener for the Start Button* ............................ 123
*Timer and Timer Event Listeners* .............................. 124
*Callback Function for Starting the Timer* ..................... 124
*Callback Functions for Timer Events* ......................... 125
Wrapping Up ......................................................... 127

**PART 2**  **EXPLORING THE BASICS OF CLASSES**

**CHAPTER 8**  **WHAT IS A CLASS?** ................................ 130
Overview of a Class ................................................... 132
Variables Revealed .................................................... 133
Creating a Class ....................................................... 134
Wrapping Up ......................................................... 138

**CHAPTER 9**  **BUILDING OUT THE CLASS** .................. 140
Parts of a Class ....................................................... 142
Package *Statement* ................................................ 142
Import *Statement* .................................................. 142
Class *Statement* .................................................... 143
Class *Constructor* .................................................. 143
Creating a Class Instance .......................................... 144
<table>
<thead>
<tr>
<th>Chapter 13</th>
<th>ADVANCED BOOLEAN LOGIC AND RANDOM NUMBERS</th>
<th>214</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Using Logic Operators</td>
<td>216</td>
</tr>
<tr>
<td></td>
<td>The AND Operator</td>
<td>217</td>
</tr>
<tr>
<td></td>
<td>The OR Operator</td>
<td>218</td>
</tr>
<tr>
<td></td>
<td>The NOT Operator</td>
<td>218</td>
</tr>
<tr>
<td></td>
<td>Building Complex Conditionals</td>
<td>219</td>
</tr>
<tr>
<td></td>
<td>Generating Random Numbers</td>
<td>221</td>
</tr>
<tr>
<td></td>
<td>Wrapping Up</td>
<td>225</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 14</th>
<th>WORKING WITH TEXT AND THE KEYBOARD</th>
<th>226</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Working with Text Fields</td>
<td>228</td>
</tr>
<tr>
<td></td>
<td>Customizing the Text Style</td>
<td>231</td>
</tr>
<tr>
<td></td>
<td>Making Changes to Style Later on</td>
<td>233</td>
</tr>
<tr>
<td></td>
<td>Creating Your Own ActionScript Fonts</td>
<td>234</td>
</tr>
<tr>
<td></td>
<td>Creating the Quiz Layout</td>
<td>238</td>
</tr>
<tr>
<td></td>
<td>Working with Escape Sequences</td>
<td>241</td>
</tr>
<tr>
<td></td>
<td>Creating the KeyboardEvent Handler</td>
<td>246</td>
</tr>
<tr>
<td></td>
<td>Differences between Code Types: Key Codes versus Character Codes</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td>Recognizing Special Keys</td>
<td>251</td>
</tr>
<tr>
<td></td>
<td>Wrapping Up</td>
<td>254</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 15</th>
<th>CREATING GROUPS OF OBJECTS AND REPEATING ACTIONS USING LOOPS</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>What Are Loops?</td>
<td>258</td>
</tr>
<tr>
<td></td>
<td>Using the for Loop</td>
<td>258</td>
</tr>
<tr>
<td></td>
<td>Controlling the Flow of Loops with break and continue</td>
<td>261</td>
</tr>
<tr>
<td></td>
<td>Nesting Loops</td>
<td>262</td>
</tr>
<tr>
<td></td>
<td>Another Style of Loops, the do Loop</td>
<td>264</td>
</tr>
</tbody>
</table>
Creating Groups of Items with Arrays .......................................................... 265
Modifying an Array ......................................................................................... 266
Using Loops to Create Arrays ..................................................................... 269
Looping through an Array ............................................................................. 274
Wrapping Up .................................................................................................... 277

PROJECT 2

DICEOUT! ......................................................................................................... 278
Project Specification: DiceOut ......................................................................... 280
Visual Design Review: DiceOut ....................................................................... 281
Kick-Off Meeting Notes: DiceOut ................................................................... 283
Solution and Walkthrough: DiceOut .............................................................. 284
Overview of the Document Class .................................................................. 284
Walkthrough of the Document Class .............................................................. 288
Wrapping Up .................................................................................................... 297

PART 4 GETTING CREATIVE WITH ACTIONSCRIPT

CHAPTER 16

DRAWING WITH ACTIONSCRIPT ................................................................... 300
Drawing and Code ............................................................................................. 302
Sprites: MovieClips without Timelines ......................................................... 302
Your First Shape ............................................................................................... 303
Extending the Sprite Class ............................................................................... 305
Drawing Lines and Working with Strokes ...................................................... 307
Drawing Ahead of the Curve .......................................................................... 312
Drawing Shapes ................................................................................................ 315
Using Fills ........................................................................................................ 318
Building Gradients .......................................................................................... 321
Looping with the Drawing API ........................................................................ 326
Wrapping Up ..................................................................................................... 330

CHAPTER 17

ANIMATION USING ACTIONSCRIPT ............................................................ 332
ActionScript Animation = Location + Time .................................................. 334
Moving an Object Using a Timer ..................................................................... 336
Creating Random Animations ........................................................................ 338
Animating Multiple Objects with Loops ........................................................ 343
Fine-Tuning Animations .............................................. 348
Wrapping Up ........................................................ 366

CHAPTER 18  WORKING WITH EXTERNAL MEDIA ............. 368
Using Images ........................................................ 370
Playing Audio from the Web ......................................... 373
Playing Video from the Web ......................................... 376
Altering the Playback of Video ...................................... 379
Pausing Video ......................................................... 382
Rewind and Seek ...................................................... 384
Wrapping Up ........................................................ 389

PART 5 CREATING MULTI-SCREEN PROJECTS

CHAPTER 19  DESKTOP APPLICATIONS WITH ADOBE AIR ....... 392
Getting Started ...................................................... 394
Working with Desktop Events ..................................... 397
Minimize .............................................................. 397
Maximize .............................................................. 398
Drag ................................................................... 401
Close ................................................................... 404
Resize .................................................................. 406
Creating a Resizable Layout ....................................... 411
Configuring Your AIR Project .................................... 419
Icons ................................................................... 419
Certificate ............................................................. 420
Publish and Install .................................................. 422
Wrapping Up ........................................................ 425

CHAPTER 20  MOBILE APPLICATIONS WITH ADOBE AIR .......... 426
Getting Started ...................................................... 428
Setting Up Your Testing Devices ............................. 429
Creating a Mobile Project ........................................ 431
Your First Android Application .................................. 431
Your First iOS Application ...................................... 434
Welcome to ActionScript. Over the next several chapters, you’ll be introduced to one of the most versatile programming languages to create web applications for the browser, desktop applications, and mobile apps for multiple platforms. For years the Flash Platform has provided people with the most powerful set of technologies to creatively express themselves across multiple screens and platforms with its combination of the Flash Player and AIR runtimes, tools like Flash Professional CS5.5 and Flash Builder 4.5, and languages and frameworks like ActionScript 3.0 and Flex 4.5.

Over the last several years, I have taught people how to make their projects interactive and how to captivate and engage users. During that time at San Francisco State University, my series on Adobe TV, and conference appearances, I have appreciated the difficulty of learning scripting and coding. Learning programming is a steep task, and there are many ways to teach it. What I have found is that combining programming basics, simple examples, problem solving, and real-world projects has been very effective, and it is what you have in your hands (or on your screen) now.

WHO THIS BOOK IS FOR

This book is crafted for people who are familiar with Flash Professional, the animation and interactive design tool from Adobe Systems that is part of Creative Suite. The lessons and projects here assume that you have a basic understanding of the Flash Professional product. This book is designed for people who are new to coding or are struggling with the migration from ActionScript 2.0 to 3.0. Here are some examples of what you should know and be able to do before attempting to start with this book:

- Import graphical assets from Creative Suite design tools
- Create timeline animations using tweens using keyframes
- Create symbols using the Library panel
- Organize and rename timelines in the Timeline panel and symbols in the Library panel
- Publish and build animations for the web browser
With these basic skills, you can create very interesting web animations; however, without ActionScript, the animations lacked any interaction with the user, and there is no way to bring them to other platforms including mobile devices. That is exactly what this book will teach you—how to make these projects interactive and take them further.

The latest edition, Flash Professional CS5.5, has added a significant number of new features to support mobile app creation that are covered at the end of the book.

WHO THIS BOOK IS NOT FOR

If you are already an intermediate or advanced coder, this book may be too basic for your needs. There are a significant number of books that focus on advanced ActionScript 3.0 concepts, including the adoption of best practices and code design patterns that will make you a better and more proficient coder.

In addition, if you have never worked with Flash Professional, I recommend you learn how to use the basic product before tackling the contents here. There are excellent books available to help you learn how to get started with Flash Professional to create animations and how to master design workflows when working with Creative Suite design applications like Photoshop, Illustrator, and Fireworks.

HOW YOU WILL LEARN

This book has a specific methodology for how the concepts are introduced. First are the fundamentals of how to interact and work with objects that are on the Stage. The examples that are in the book are simple—and this is intentional, to help you understand how ActionScript works without getting into the weeds of your project’s design or assets. You can adapt and expand these simple examples for your own projects.

After you gather a sizable amount of new ActionScript know-how, it is time to put it to work. There are three major projects in the book that pose real-world situations for you to solve using the skills you have learned. The projects present you with a programming challenge and ask you to solve it. You can compare your finished projects with the examples in the book to discover how your approach matches or differs.
WHAT YOU WILL LEARN

This book is divided into five major parts.

PART 1: GETTING THE FUNDAMENTALS
You’ll learn general ActionScript concepts that you can use to make ActionScript interact with objects on the Stage and in the Library of your project. You’ll build on this, understanding how to flow your code through reusable modules called functions, and then how to respond to user interaction with event handlers.

PART 2: EXPLORING THE BASICS OF CLASSES
You’ll jump into the basics of what is called object-oriented programming (OOP), which is what separates the coders from the scripters. Through OOP you can unlock a lot of flexibility in how you create projects, learning how to make reusable objects and containers that can extend the sophistication of your projects.

PART 3: RESPONDING TO CONDITIONALS AND WORKING WITH LOGIC
Adapting your project based on certain conditions then is the focus of the next section, where through the use of conditionals, your project can adapt to different interactions from the user or even to random events to begin introducing gaming concepts to your project.

PART 4: GETTING CREATIVE WITH ACTIONSCRIPT
Although ActionScript is a programming language, it has its creative side. This is covered in the fourth section, where you will learn how to draw, animate, and work with external assets in your projects.

PART 5: CREATING MULTI-SCREEN PROJECTS
After you have mastered all the previous topics, it is time to take your projects out of the browser and take advantage of the Flash Platform to create desktop applications for Windows and Mac OS X operating systems and mobile apps for the popular Android and iOS platforms.

You’ll cover a lot, but at the end, you’ll have a solid foundation on how ActionScript works and the power that you have at your fingertips to express yourself across screens and platforms.

So let’s get started!
ActionScript 3.0 is the programming language of the Adobe Flash Platform, a multi-screen and multi-device development platform for creating interactive and expressive content. With the latest generation of the Adobe runtimes, Flash Player and AIR, you can take your ideas and creative vision to the browser, desktop, mobile phones, tablets, and Internet-enabled televisions. Let’s review some of the tools that you’ll be working with.

THE TOOLS AND RUNTIMES

In the course of this book, there are three main tools and runtimes that you’ll be working with:

FLASH
PROFESSIONAL CS5.5
The latest generation of the Flash authoring tool combines powerful animation capabilities, library management, and an integrated coding environment designed for ActionScript 3.0 coding. Part of Creative Suite 5.5, Flash Professional CS5.5 adds new support to work with the latest generation of Adobe AIR and Flash Player 10.2 to create content and applications for the popular Android and iOS mobile platforms.

FLASH PLAYER 10.2
Flash Player is what brings the web to life. It is the Internet plug-in for your desktop or mobile phone that allows you to play interactive content, video, and games. The latest version includes enhanced support for hardware acceleration, better video playback, and memory and processor performance optimization.

ADOBE AIR 2.6
The Adobe AIR runtime is what allows interactive designers and developers to take their applications outside the browser and bring them to the Windows and Mac OS X operating systems as desktop applications, or to the Android and iOS platforms as installable mobile applications.
OTHER HELPFUL TOOLS

Although not part of this book, there are other tools that are helpful for working with the Flash Platform, including:

ADOBE FLASH BUILDER 4.5
Flash Builder is the professional coding IDE for the Flash Platform. It includes advanced programming functionality to optimize projects, and it makes working with larger projects and coordinating projects with teams easier. Flash Builder also supports working with Flash Professional projects and using the Adobe Flex framework.

ADOBE FLEX 4.5
The Flex framework is used specifically to create data-driven applications for the browser, desktop, and mobile devices. Incorporating skinnable components, declarative layout, ActionScript logic, and support for a growing set of platforms, it is the fastest way to create a robust application for multiple screens and devices.

ADOBE FLASH CATALYST CS5.5
Flash Catalyst is designed to work in a team environment when a designer and a developer are building an Internet application using the Flex framework. Interaction designers can create skins for Flex components and craft the overall user interface of a Flex application as a wireframe, prototype, or a finished application. Flash Catalyst CS5.5 introduces round-trip functionality with Flash Builder 4.5 to allow designers and developers to work collaboratively.
4

ACTIONSCRIPT
AND MATH
ActionScript has tons of mathematical operators built in to the language to help you evaluate mathematical equations. Now, I wouldn’t throw away your handheld calculator just yet. ActionScript has a lot of power, but it’s designed to help with your applications, not for general use. In addition to these mathematical operators, there are some functions that can help with common mathematical tasks like rounding numbers.

In this section, you’ll learn all the basic arithmetic operators that you’ll use in ActionScript. Also, there are some convenient shortcuts to make working with math easier that you’ll cover as well.
In ActionScript, you can use simple math operators to perform arithmetic functions with your numbers or variables. The math functions that are part of ActionScript are nearly identical to basic math functions that you already know. Some of the symbols and names are different, but the principles are the same.

**ADDITION AND SUBTRACTION**

Let’s get started with adding and subtracting numbers.

1. Create a new ActionScript 3.0 project in Flash Professional CS5.5 and enter the following code into the timeline:

   ```actionscript
   // Math operators: addition and subtraction
   trace ( 2 + 3 );
   trace ( 3 - 2 );
   ```

2. Run the project and look at the Output panel; you’ll see the following:

   5
   1

   This shouldn’t be surprising, since you are adding and subtracting the numbers. You use the + and – operators to indicate that you are adding or subtracting. One thing to note is the white-space characters used in the example. Notice the spaces between the operators and the numbers. This is for readability and doesn’t affect the execution of the code. You can remove the spaces if you want, for example:

   ```actionscript
   trace ( 2 + 3 );
   trace (2+3);
   ```

   These two lines perform exactly the same function and will generate the same result.
ADDITION OR CONCATENATION?

In the previous chapter, you used the + sign, but it wasn't a mathematical operator. You can use the + operator to do two things. When working with strings, the + operator is called the concatenation operator and takes two strings and combines them together, in essence gluing the end of one string to the beginning of the next. When working with numbers, the + operator is the addition mathematical operator, adding two numeric values together and generating a new numeric result.

Look at the following example.

1. Remove the existing code and enter the following code:

```javascript
// Addition vs. Concatenation
trace ( 2 + 2 ); // addition
trace ( "two" + "two" ); // concatenation
trace ( "2" + "2" ); // concatenation
```

2. Run this code; you'll see the following displayed in the Output panel:

```
4
twotwo
22
```

The first line of code in the example is pretty simple; you are adding the numbers 2 and 2 using the addition operator, resulting in a value of 4.

The second line of code has two strings, denoted by quotation marks, that are being "glued" together, creating a single string using the string concatenation operator. The result is “twotwo.”

The last line uses the number 2 on both sides of the operator. Notice that the numbers are surrounded by quotation marks, which means that it is no longer a number value, but instead the character 2. When you force the number 2 to be a string using quotation marks, the + operator concatenates the strings, “gluing” them together forming the string, 22.
What makes this confusing is that the Output panel doesn't distinguish between strings and numbers. So, when you see 22 in the Output panel, is it a number or a string? There is a way to find out the type of a value and display it: by using the `typeof` statement.

3. To see how `typeof` works, update the previous example as follows:

```ActionScript
// Addition vs. Concatenation
trace ( typeof(2 + 2) ); // addition
trace ( typeof("two" + "two") ); // concatenation
trace ( typeof("2" + "2") ); // concatenation
```

4. Run this updated example; you'll see the following in the Output panel:

```text
number
string
string
```

What is happening is that the operation (either addition or concatenation) is taking place, and the `typeof` statement is determining the type of the result and then sending that to the Output panel via the `trace` statement.

Now for one final twist. If you mix up the number and string types, what happens?

5. Replace the existing code with the following:

```ActionScript
trace ( 2 + "2" );
```

Wow. Now you have a number on the left side, and a string on the right side. Who wins?

6. Run the project.

The answer is that the string wins. The result is the string, “22”. In this case the operator converts the number 2 to the string “2” and then “glues” it to the right “2” creating the string “22”. It seems confusing at first, but after you work with it a while, it will become second nature to you—promise!
MULTIPLICATION AND DIVISION

Now, look at the * and / operators for multiplication and division.

1. Replace the code in the timeline with the following:

```javascript
// Math operators: Multiplication and Division
trace ( 2 * 3 );
trace ( 5 / 2 );
```

The first statement uses the multiplication operator, which is an asterisk, `*`. The division operator is a forward slash, `/`, and the order of the division is that it divides the value on the left by the value on the right.

2. Run the project; you’ll see the following in the Output panel:

```
6
2.5
```

Again, pretty simple stuff—but the next one will probably be new to you.

MODULO, THE OPERATOR FORMERLY KNOWN AS LONG DIVISION WITH REMAINDERS

The modulo operator finds the remainder after a division operation. The modulo is quite helpful in many situations, including determining if a number is odd or even. Take a look at how it works.

1. Replace the code you have with the following, and take a look at the output:

```javascript
// Math operators: Modulo
trace ( 5 % 2 );
```

2. Run the project; you’ll see the following displayed in the Output panel:

```
1
```

The `%` symbol invokes the modulo operator, finding the remainder after attempting a division of the value on the left with the value on the right. In this example, it divides 5 by 2, resulting in 2 and a remainder of 1. To see this written out in long division format, check out Figure 4.1.

![Figure 4.1](image.png)
What on Earth Is Modulo Used For?

That is a great question, and one that has a great answer as well. One of the most common uses is to determine if a value is a multiple of another. For example, to find out if a value is an even multiple of 3, you can use something like this:

```
myValue % 3;
```

If the result is 0, that means there are no remainders, and the number is an even multiple of 3.

Another common use is to determine if a number is even or odd. Even numbers are evenly divisible by 2, so by that definition you could use this:

```
myValue % 2;
```

If the result is 0, the number is evenly divisible by 2, making it even. If it isn't, then the number is odd.
VARIABLES AND COMBINED ASSIGNMENT OPERATORS

You’ll commonly want to complete a math function and assign the resulting value back to some named object, called a variable. ActionScript makes this easier by letting you combine arithmetic and assignment operators together. Take a look at an assignment operator example:

1. Create a new ActionScript 3.0 project and enter in the following code for the project:

```actionscript
// Assignment Operators
var myValue:Number = 2;
myValue = myValue + 2;
trace(myValue);
var myOtherValue:Number = 2;
myOtherValue += 2;
trace(myOtherValue);
```

2. Run the project. You’ll get the following in the Output panel:

```
4
4
```

Let’s walk through the code and explain how you get this result and what role variables and combined assignment operators play.
**VARIABLES**

You haven’t really seen much about the `var` statement yet, so let’s reveal a little bit more about it. You have used it in the past to create named object containers that you have then assigned MovieClip symbols to using the `new` statement. You can also use `var` to create variables; in fact, variables is what `var` stands for. Variables are named objects that can contain variable values.

Take a look at the second line of the assignment operators example:

```actionscript
var myValue:Number = 2;
```

The `var` statement is creating a variable called `myValue`. See that `:Number` after the variable name? You have to tell ActionScript what type of data your variable can hold, similar to how you did when using the `function` statement. In this case, you are saying that `myValue` will contain a number. When you create the variable, it is empty, but when you assign the numeric value 2 to it, you can refer to that value using the name `myValue`.

```actionscript
myValue = myValue + 2;
```

```actionscript
trace(myValue);
```

On the second line above, you are accessing the `myValue` object and are assigning a new value to it. Notice that you are not using the `var` statement here, because `var` is only used to create a new variable. You don’t need to use it again if you are referring to a variable that has already been created. Before you assign the value, you need to complete the evaluation on the right side of the assignment operator. In this case, you are taking the existing value of `myValue`, 2, and adding the value 2 to it. This value is then assigned back to `myValue`, overwriting the existing value. In the last line of the first block, you send that value to the Output panel using the `trace` statement, which displays 4.

This completes the analysis of the first part of the code.
COMBINED ASSIGNMENT OPERATORS

Take a look at the second block of code. This section of code works identically to the first block, with two exceptions. In this section, you are creating a new variable called myOtherValue:

```javascript
var myOtherValue:Number = 2;
myOtherValue += 2;
trace(myOtherValue);
```

In the first line, you need to use the var statement since you have not created that variable before. You then assign the numeric value 2 to it.

On the next line, you come across the first combined assignment operator, `+=`. This operator is combining addition with assignment. In this case it is taking the existing value of `myOtherValue` and is adding 2 to it and automatically assigning it back to the `myOtherValue` variable. Always put the arithmetic operator before the assignment operator. You can use this shortcut with any of the basic arithmetic operators:

```javascript
// All combined assignment operators
var myValue:Number = 100;
myValue += 50; // 100+50 = 150
myValue -= 125 // 150-125 = 25
myValue *= 3   // 25*3 = 75
myValue /= 5   // 75/5 = 15
myValue %= 4   // 15%4 = 3
trace (myValue);
```

Programmers often use these combined assignment operators as shortcuts since they are nice time savers. Hopefully, you’ll find they are too!
When you work with ActionScript a lot, you’ll commonly be adding or removing 1 from variables and properties.

To make this process easier, there is a shortcut called the increment and decrement operators. Take a look at the following code.

1. Create a new ActionScript 3.0 project and enter in the following code for the project:

```actionscript
// Increment and Decrement
var myValue:Number = 5;
trace(myValue);
myValue++;
trace(myValue);
myValue--;
trace(myValue);
```

2. Run this project. You’ll see the following in the Output panel:

```
5
6
5
```

In the increment and decrement example, the value of myValue is initially set at 5 and is sent to the Output panel. The number is then increased by 1 and sent again, resulting in 6.

When you add a double minus, --, to the end, it decrements the value by 1. The value of myValue is already 6 based on the previous function, and is then decremented to be 5 again.
When you add a double plus, ++, to the end of a variable name, you increment it by 1. As a result, the following three lines of code do the exact same thing:

```
myValue = myValue + 1;
myValue += 1;
myValue++;
```

The following three lines of code do the same thing, similarly to the example earlier for the increment operator:

```
myValue = myValue - 1;
myValue -= 1;
myValue--;
By default, mathematical functions do not run from left to right, but follow a specific order of operations. You may recall from math classes that certain mathematical functions are calculated before others, regardless of their left-to-right order.

1. Create a new ActionScript 3.0 project and enter in the following code for the project:

   ```actionscript
   // Order of Operations
   var answer:Number = 2 + 3 * 2 / 4 - 1;
   trace(answer);
   ```

   In this example, you have a number of math functions that are running from left to right. If you don’t follow the order of operations and evaluate it from left to right, you get 1.5, as shown in Figure 4.2.

2. Run the code. You’ll see what might seem unexpected: 2.5. Why? Because certain math functions are executed before others. In fact, this is the order:

   1. Multiplication, Division, and Modulo
   2. Addition and Subtraction
All the multiplication, division, and modulo operations are processed from left to right to the end. Then calculation starts again from the left and processes addition and subtraction. Look at Figure 4.3 to see how this works.

When the Flash runtime looks at the ActionScript, it starts from the left, evaluating the expression:

- It ignores the 2 + 3, since the rules dictate processing only multiplication, division, and modulo at this point.

  - $3 \times 2 = 6$ ①
  - $6 / 4 = 1.5$ ②

  Since there are no more multiplication, division, or modulo operations, it returns to the beginning and processes addition and subtraction.

  - $2 + 1.5 = 3.5$ ③
  - $3.5 - 1 = 2.5$ ④

You have the final result, 2.5 ⑤, which is then sent to the Output panel.

You can alter the order of operation by using parentheses. This will force Flash to adopt a specific path of calculating the results. You’ll learn about overriding the order of operation rules in the next section.
You can force the earlier example to follow the order of operation that results in the value of 1.5. You can use parentheses to group calculations together. In the order of operations, math operations that are grouped within a pair of parentheses are always calculated first.

You can adjust the example to get the 1.5 that you originally calculated by performing the calculations from left to right:

```javascript
// Order of Operations
var answer:Number = (2 + 3) * 2 / 4 - 1;
trace(answer);
```

Now instead of skipping the first addition action, the Flash runtime calculates what is inside the parentheses first and then continues across, as shown in Figure 4.4.

When the Flash runtime looks at the ActionScript, it starts with the first set of parentheses it finds:

- \(2 + 3 = 5\), which is the only set of parentheses ①

It then starts back at the beginning with multiplication, division, and modulo:

- \(5 \times 2 = 10\) ②
- \(10 / 4 = 2.5\) ③

Now that it is finished with multiplication, division, and modulo, it starts back on the left and evaluates addition and subtraction:

- \(2.5 - 1 = 1.5\) ④

### Figure 4.4 Forcing the order with parentheses using order of operations

\[
\begin{align*}
(2 + 3) & \times 2 / 4 - 1 \\
5 & \times 2 / 4 - 1 \\
10 & / 4 - 1 \\
2.5 & - 1 \\
1.5 & 
\end{align*}
\]
You end up with 1.5 $t$, which is then sent to the Output panel.
You can nest parentheses within each other, but just make sure that every opening parenthesis has a matching closing parenthesis. This is one of the most common bugs you’ll find in your programs, unmatched parentheses and braces.
You have covered a lot of math in this chapter, but more importantly, you were able to expand your knowledge of working with numbers and variables and start doing some calculations with them. Table 4.1 will serve as a handy reference for the operations that were covered in this chapter:

**TABLE 4.1 Mathematical Operators**

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>DEFINITION</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
<td>4 + 5 results in 9</td>
</tr>
<tr>
<td>–</td>
<td>Subtraction</td>
<td>5 – 4 results in 1</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>2 * 3 results in 6</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>5 / 2 results in 2.5</td>
</tr>
<tr>
<td>%</td>
<td>Modulo</td>
<td>5 / 2 results in 1</td>
</tr>
<tr>
<td>+=</td>
<td>Addition assignment</td>
<td>if x is 5, x += 3 changes x to 8</td>
</tr>
<tr>
<td>-=</td>
<td>Subtraction assignment</td>
<td>if x is 5, x -= 2 changes x to 3</td>
</tr>
<tr>
<td>*=</td>
<td>Multiplication assignment</td>
<td>if x is 5, x *= 3 changes x to 15</td>
</tr>
<tr>
<td>/=</td>
<td>Division assignment</td>
<td>if x is 5, x /= 3 changes x to 2.5</td>
</tr>
<tr>
<td>%=</td>
<td>Modulo assignment</td>
<td>if x is 5, x %= 2 changes x to 1</td>
</tr>
<tr>
<td>++</td>
<td>Increment</td>
<td>if x is 5, x++ changes x to 6</td>
</tr>
<tr>
<td>--</td>
<td>Decrement</td>
<td>if x is 5, x-- changes x to 4</td>
</tr>
</tbody>
</table>
In this chapter, you learned the basics of working with variables and how to change numeric values using arithmetic operators in ActionScript. You also learned some of the common shortcuts advanced programmers use to save time when working with math operators, including working with combined assignment operators and the increment and decrement operators.

When working with operators in ActionScript, keep the following in mind to avoid common pitfalls and errors:

- When creating a variable and referring to it the first time, you need to use the `var` statement to create it. You can then refer to it without the `var` statement afterwards.
- When using the `+` operator, be sure to not inadvertently mix up strings and numbers, as strings will concatenate and ignore the numeric values.
- The modulo operator calculates the remainder after attempting to complete an even division.
- If you are working with a combination of multiplicative (multiplication, division, or modulo) functions and summation (addition or subtraction) functions, remember that ActionScript will evaluate your equation using mathematical order of operations.
- To quickly modify an existing value based on a function, you can use combined assignment operators to save time.
- If you are adding or subtracting 1 to or from a value, you can use increment or decrement operators, using `++` or `--` as a quick shortcut.
- To force the order of operations to do something specific, you can use parentheses to group evaluations you want to process first.
INDEX

SYMBOLS
− operator, 50
!= statement, 195
|| operator, 218
++ operator, 59
+ operator, 44–45, 50–52
= operator, 14–15
== statement, 194
! prefix, 218
& operator, 217–218
* operator, 53
% modulo operator

NUMBERS
1049 error, 417
1084 error, 46

A
accepting values in functions, 39–43
access
Keychain, 434–435, 510–511
to methods from outside classes, 153
to objects. See access to objects
passwords granting. See passwords
private statement restricting, 161–162
to values of properties, 164
access to objects
introduction to, 3
sending messages to output panel, 8–11
on Stage, 5–7
summary of, 19
Actions panel
accepting values in functions, 39–42
code completion assistance in, 153
creating new objects in, 25–26
disabled, 135
display stacks in, 29–30
dynamically adding objects to Stage, 26
graphics without timelines vs., 302
in mouse event handler creation, 71
parameters of objects in, 15–16
in timeline scripting with mouse
events, 86
writing ActionScript using,
introduction to, 9–11
ActionScript overview. See also specific
topics
accessing and manipulating objects, 3
Actions panel, 9–10
ActionScript 3.0, defined, xvi
of advanced Boolean logic, 215
of animation, 333
of building out classes, 141
of classes, generally, 131
of classes in streamlined interfaces, 157
of conditionals, 191
converting objects to MovieClips for,
5–7
of desktop applications, 393
design, 301
dynamically adding objects from
Library, 21
of event creation, 67
of external media, 369
of frame events, 99
of functions, 35
groups of objects, 257
instance names in, 7
interactivity for items on Stage, 3
of keyboards, working with, 227
of mathematical operators, 49
methodology for learning, xiv
of mobile applications, 427
MovieClip properties, 14–19
new project for, 4–10
of organizing classes, 175
Output panel, 10
prerequisites for using, xiii–xvi
of random numbers, 215
of repeating actions, 257
setup, 119–121
summary of, xv
of text, working with, 227
of timeline scripting with mouse
events, 79
of timer events, 99
versions of, 4
addChild statement, 29–30
addEventListener, 71, 75
addition operator, 50–52
Adobe AIR 2.6, defined. See also Adobe
AIR for desktop applications;
Adobe AIR for mobile applications,
xvi
Adobe AIR 2.7, defined, 438
Adobe AIR for desktop applications
certificates in, 420–422
close method in, 404–406
drag API in, 419–420
drag API in, 404–403
events in, generally, 397
icons in, 419–420
installing applications in, 422–424
of animation, 333
of building out classes, 141
of classes, generally, 131
of classes in streamlined interfaces, 157
of conditionals, 191
converting objects to MovieClips for,
5–7
of desktop applications, 393
design, 301
dynamically adding objects from
Library, 21
of event creation, 67
of external media, 369
of frame events, 99
of functions, 35
groups of objects, 257
instance names in, 7
interactivity for items on Stage, 3
of keyboards, working with, 227
of mathematical operators, 49
methodology for learning, xiv
of mobile applications, 427
MovieClip properties, 14–19
new project for, 4–10
of organizing classes, 175
Output panel, 10
prerequisites for using, xiii–xvi
of random numbers, 215
of repeating actions, 257
setup, 119–121
summary of, xv
of text, working with, 227
of timeline scripting with mouse
events, 79
of timer events, 99
versions of, 4
addChild statement, 29–30
addEventListener, 71, 75
addition operator, 50–52
Adobe AIR 2.6, defined. See also Adobe
AIR for desktop applications;
Adobe AIR for mobile applications,
xvi
Adobe AIR 2.7, defined, 438
Adobe AIR for desktop applications
certificates in, 420–422
close method in, 404–406
drag API in, 419–420
drag API in, 404–403
events in, generally, 397
icons in, 419–420
installing applications in, 422–424
introduction to, 393–394
maximize control in, 398–401
minimize control in, 397–398
publishing projects in, 423
resizable layouts in, 411–418
resizing windows in, 406–410
as runtime. See AIR runtime
starting projects in, 394–396
summary of, 425
Adobe AIR for mobile applications
Android applications, creating first,
431–434
debugging, on Android, 458–459
debugging, on iOS, 460–462
debugging, over USB generally, 458
dragging with touch in, 442–444
gesture events, generally, 450
interactions for devices, generally, 439
introduction to, 427
iOS applications, creating first, 434–438
long press events in, 445–449
optimizing content, 463–464
pinch to zoom gestures, 450–455
projects, creating new, 431–438
projects, setup for, 428–430
rotate gestures in, 455–457
as runtime. See AIR runtime
summary of, 465
tap interactions in, 439–442
testing devices in, 429–430
touch interactions in, 439–442
Adobe Fireworks, 419–420, 471
Adobe Flash Builder 4.5, xvii
Adobe Flash Catalyst CS5.5, xvii
Adobe Flash Platform
Adobe Flash Builder 4.5 for working
with, xvii
Adobe Flash Catalyst CS5.5, xvii
Adobe Flex 4.5 for working with, xvii
AIR for mobile applications on, 427
defined, xvi
resources for publishing AIR
applications on, 423
time of. See Flash runtime
tools of, xvi
Adobe Flash Player
defined, xvi
object parameters in, 16
Times New Roman font in, 231
trace statements not appearing in, 10
visible property in, 18
Adobe Flex 4.5, xvii
advanced Boolean logic. See Boolean logic, advanced
AIR runtime. See also Adobe AIR for desktop applications; Adobe AIR for mobile applications
of AIR 2.7, 438
creating desktop applications with, generally, 393, 425
creating mobile applications with, generally, 427, 465
downloading, 394, 423
introduction to, xvi
iOS vs. Android in, 428
align property, 406
alpha property
of fills, 320
of gradients, 321, 323
in starfield project, 352–354, 365
of strokes, 308
analog clock appearance, 114–115
AND operator, 217–218
Android 2.2, 427, 429
Android 3.x (Honeycomb), 427
Android devices, 430, 504–505
Android Package files (.APK), 428
Android platform
applications for, generally, 428
creating first application for, 431–434
debugging on, 458–459
Flipr game on. See Flipr game project
introduction to, 427
testing applications for, 504–505
touch interactions in, generally, 439
website publishing Air for, 503
animation
adding more visual elements to, 358–361
brightness adjustments for, 352–355
fluidity of, 355–358
intervals for, 355–358
introduction to, 333
location on screen and rate of change in, 348–351
location + time as, generally, 334–335
of multiple objects with loops, 343–348
random, 338–342
rate of change of, 348–351, 355–358
scaling size of visual elements in, 361–365
summary of, 366
timer-based, 110–111, 336–338
APIs (application programming interfaces)
drag, 401–403
for dragging windows in AIR, 401–403
FLVPlayback, 376–382
for graphics. See graphics API
(application programming interface)
for mobile applications in AIR, 439
.APK files (for Android Packages), 428
Apple Devices Application files (.IPA), 428
Apple Mac OS X
AIR applications for, 424
Applications folder in, 420
desktop applications for, 394
Snow Leopard, 504
Apple mobile devices
introduction to, 427–429
resolution for, 434
testing applications on, 506
application programming interfaces (APIs). See APIs (application programming interfaces)
arithmetic functions. See mathematical operators
arrays
animation in, 342–345
creating groups of items with, generally, 265–266
creating with loops, 269–273
defined, 257
for game board creation, 489
introduction to, 257
looping through, 274–276
loops for, 258
modifying, 266–269
removing items from, 268–269
summary of, 277
values of Flipr tiles in, 489
arrow keys, 251–253
artwork in projects. See drawing
.as file extension, 144, 154
assignment operator
assigning new objects to named objects with, 33
defined, 14
on Output panel, 55
audio, 373–375
authorTime, 21
auto-generated imports, 121
B
Basic Latin character range, 235
begin method, 318–320
beginGradientFill method, 321
Bondini font, 472
Boolean conditional tests, 258
Boolean logic, advanced. See also Boolean variables
complex conditionals, building, 219–220
drag and drop for mouse, 198–200
equality, demonstrating, 196–198
equality, testing for, 194
inequality, demonstrating, 196–198
inequality, testing for, 195–196
introduction to, 192–193
break statement, 261–262
brightness adjustments, 352–355
Brimelow, Lee, 508
broadcast-listener relationships. See also event listeners, 68
building out classes. See classes buttons
for AIR desktop applications, 395–396
close windows, 404–406
drag windows, 401–403
event callback objects for, 87–88
for Flipr, 472, 480–481
forwardButton, 83, 90
goBackButton, 472, 481
graphics API and, 374, 380, 396
hoverButton, 87–94
instructionsButton, 472, 481
internal timeline for special effects of, 85–86
labels for, 382–384
layer of Stage, 97
layout of, 411–418
maximize, 398–401
minimize, 397–398
object type, 22
padding around, 413
playButton, 85–86, 87–93
resize windows, 406–410
start, 123
startGameButton, 472, 474
stopButton, 81–82, 90–93
toggle, 382–384
video playback, 382–384
x and y coordinates of, 412–413
Cache as Bitmap object setting, 475

CacheAsBitmap property, 463

callback functions
  _endSplash, 480
  arrays and, 272–273, 276
circleClick, 71–73
event callback objects, 89–95
event handlers
  _endSplash, 480
event loop, 276, 364
for mobile applications, 444
getters and setters, 164–165
for touch interactions in mobile applications, 444
for starting timers, 124–125
for timer events, 125–126
for touch interactions in mobile applications, 444
for timer events, 125–126
in flash.display.SimpleButton class, 141
introduction to, 69
keyboardEvent handler and, 246–249
in minimize window controls, 398
in streamlining interfaces, generally, 157–173
for touch interactions in mobile applications, 444
in streamlining interfaces, generally, 157–173

callouts, 37–38, 41
camel case, 7
capitalization, 10
Captivate, 429
cell method, 224
cell-based animation, 334
cell phones, 429–430
centering objects, 451–452
Certificate field, 434–435
Certificate Signing Requests, 509
Certificate Signing Settings window, 420
cellphoneClick method, 71–73
circles, 318
circular geometry, 225
class-based scripts, defined, 8
class constructors. See also classes
cell-based animation, 318
cell phones, 429–430
circleClick callback function, 71–73
circles, 318
circular geometry, 225
class-based scripts, defined, 8
class constructors. See also classes
  button label customizing with, 146–147
class instance creation using, 144–145
introduction to, 143
multiple properties, customizing with, 148–149
parameters of, adding, 146–149
parameters of, generally, 146
parameters of, making optional, 150–151
classes
  building out, generally, 141, 154
  constructor parameters in,
    generally, 146
  constructors of, generally, 143
  containers for, 133
  creating, 134–137
  definition warning, 24
  Document. See Document class
  extending functionality of, 305–307
  finding definitions for, 137
  flash.text.TextField, 229
  flash.text.TextFormat, 231
  fl.video class package, 378
  GameBoard, 482–488
  getter methods in, creating, 165–166
  getter methods in, generally, 164
  getter methods in, using, 166–168
  instances of, creating, 144–145
  introduction to, 131
  Math, 221–224
  methods in, accessing from outside classes, 153
  methods in, creating generally, 152
  optional constructor parameters in,
    150–151
  organizing, generally, 173, 187
  overview of, 132
  package folders of, creating, 176–180
  packages for, changing source paths of, 183–186
  packages for, creating nested, 182–183
  packages for, referring to all classes in, 180–182
  parts of, 142
  polymorphism of, 164
  private, best practices for naming, 163
  private, generally, 161–162
  ProLoader class, 370–371, 389
  properties of, customizing multiple, 148–149
  public, best practices, 164
  public, generally, 158–160
  setter methods in, creating, 165–166
  setter methods in, generally, 164
  setter methods in, using, 166–168
  Sound class, 374–375, 389
  Sprite, 302–307
  in streamlining interfaces, generally,
    157, 173
  introduction to, 131
  summary of, 128
  TextFormat, 231–232
  TouchGestures Document class, 450
  variables in, generally, 133
  VideoButton class, 382–384
  clickedTile property, 500
  _clickTile method, 499
close method, 404–406
close window control, 411–413
code-signing certificates, 422–423
color
  of Flipr tiles, 468–469, 500
  hexadecimal numbers for, 231–232
coding out, 29
called assignment operators, 55, 57
commercial code-signing certificates, 422–423
Components panel, 376
commenting out, 29
concatenation, 44–45, 51–52
components
 添加 with, 200–201
  for animation, 345–348
Conditional elements
  adding feedback with, 200–201
  for animation, 345–348
  Boolean variables as, generally, 192–193
  drag and drop with, 198–200
  equality, demonstrating, 196–198
  equality, testing for, 194
  if statements, 204–206
  if.else if statements, 209–212
  if.else statements, 206–208
  inequality, demonstrating, 196–198
  inequality, testing for, 195–196
  introduction to, 191
  introduction to, 191
  summary of, 213
testing conditions with, generally, 202–204
  testing conditions. See testing conditions
  for, generally, 198–200
  equality, demonstrating, 196–198
  equality, testing for, 194
  if statements, 204–206
  if.else if statements, 209–212
  if.else statements, 206–208
  inequality, demonstrating, 196–198
  inequality, testing for, 195–196
  introduction to, 191
  summary of, 213
  testing conditions with, generally, 202–204
  testing conditions. See testing conditions
confirmation messages
AIR application publication, 422
in array creation, 273
trace statement generating, 170
const statement, 413–417
constant values, 413–417
constructor parameters. See also class constructors
customizing button labels with, 146–147
customizing multiple properties with, 148–149
making optional, 150–151
constructors. See class constructors
containers
for classes, 133, 138
for objects, 470
variables as, 13
continue statement, 261–262
controls
close window, 404–406, 411–413
drag window, 401–403, 411, 413
Flipr main, 480–482
Flipr player, 494–500
location of buttons for window, 411–413
maximize window, 398–401, 411–412
minimize window, 397–398, 411, 418
native window, 395–397, 401, 409
resizable layout of window, 415–417
resize window, 406–410
values in window, 413–417
y coordinate of window, 412–413
Convert to Symbol dialog box, 5, 23
countdown clock project
ActionScript setup in, 119–121
auto-generated imports in, 121
display objects in, 122–123
event listeners for start buttons in, 123
Flash Professional, setup in, 117–118
kick-off meeting notes for, 116
solution and walkthrough of, 117–126
specifications for, 114
summary of, 127
timers, callback functions for events in, 125–126
timers, callback functions for starting, 124–125
timers, timer event listeners for, 124
visual design review of, 115
createGradientBox method, 324
curly brace locations, 38, 46
currentCount property, 470
curves, 312–315
curveTo method, 312–317
D
data types, 40
debugging
on Android, 458–459
on iOS, 460–462
over USB, generally, 458
decimals, 221–225
decrement operators, 58
defaultTextFormat property, 233
defining variables, 159
degrees, 325
Dell mobile devices
applications for, generally, 428
testing applications with, 429, 505
Deployment Settings, for iOS, 434–435
desktop applications
AIR for, introduction to, 393–394
certificates for, 420–422
configuring projects for, generally, 419
desktop events in, generally, 397
drag API in, 401–403
icons in, 419–420
installing, 422–424
maximize control in, 398–401
minimize control in, 397–398
publishing projects in, 422
resizable layouts in, 411–418
resizing windows in, 406–410
starting projects in, 394–396
summary of, 425
desktop events. See also desktop applications
close, 404–406
drag, 401–403
introduction to, 397
maximize, 398–401
minimize, 397–398
resizing windows in, 406–410
Developer Member Center of iOS, 507
Developer Program for iOS, 506
Development Certificates, 510–511
DiceOut game project
Document class overview, 284–288
Document class walkthrough, code for rolling dice, 292–295
Document class walkthrough, contents before constructor, 288–290
Document class walkthrough, ending game, 296
Document class walkthrough, event listeners, 290–292
Document class walkthrough, scores, 296
introduction to, 279
kick-off meeting notes on, 283
solution and walkthrough, generally, 284
specifications for, 280
summary of, 297
visual design review of, 281–282
display objects, 122–123
Display section of Properties panel, 463
display stacks
defined, 25
instances in, assigning properties to, 30
objects in, creating new, 25–28
objects in, dynamically adding to Stage, 29–30
displayInstructions method, 481
displayState property, 401
displayTile method, 498
division operator, 53–54
do loop, 264
Document class
for AIR desktop applications, starting, 396
for animation. See animation arrays in, 265
Booleans, 192–193, 197–209
creating, 169–171
in DiceOut game project, overview, 284–288
in DiceOut game project, walkthrough, 288–296
in Flipr game project, 475–478
for loop for, 259
in game board creation, 488–489
initialization methods in, 171–172
for iOS applications, 440–449
for minimize control, 397
nested packages for, 182–183
newGame method in, 494–496
packages for, creating, 176–180
packages for, referring to all classes in, 180–182
for pausing video, 382–383
PlayAudio project and, 373–375
for playing video from Web, 377–378
ProLoader class and, 370–372
RandomNumber, 221–224
for seeking in videos, 385–387
source path for, 183–186
sprites. See sprites
TextFields, creating, 228–230
Document class (continued)

TextFields, creating KeyboardEvent handler in, 246
TextFields, customizing style with, 231–237
TextFields, layout with, 238–245
TouchGestures, 450–456
for video playback alteration, 380–381
dot notation, 15
drag and drop
for desktop applications, 401–403
drag window control for, 411, 413
for mobile applications, 442–444, 447
testing for, 218
testing location of objects, 196–198
touch interactions for, 442–444
using mouse, 198–200
drawCircle method, 318
drawEllipse method, 318
drawing
circles, 318
code for, generally, 302
curved, 312–315
degrees in, 325
ellipses, 318
fills, 318–320
gradients, 321–326
introduction to, 301
lines, drawing, 307–312
loops and, 326–329
rectangles, 317
rounded rectangles, 317–318
shapes, first, 303–304
shapes, generally, 315–317
Sprite class, extending, 305–307
Sprite class, generally, 302–303
strokes for, 307–312
summary of, 330
drawRect method, 317
drawRoundRect method, 317
Droid, 429
drop and drag. See drag and drop
dropDrag method, 200
dynamic text labels, 146
dynamically adding objects to Stage
comments, creating, 31–32
display stacks, assigning properties to
instances, 30
display stacks, creating new objects
for, 25–28
display stacks, defined, 25
display stacks, working with, 29–30
introduction to, 21
dynamically created instances, 30
dynamically created instances, assigning
properties to, 30
dynamically loading images, 370

e
event object, 73, 89–95
editing-in-place, 85
ellipses, 318
embedded fonts, 234–237
embedded video, 376
emulators, 430
endFill method, 320
endGameButton, 472, 474
_endSplash callback method, 480
endTouch callback function, 449
ENTER_FRAME event, 104–107
equality, 194, 196–198
equality, 194, 196–198
errors
1049 error, 417
1084 error, 46
class definition warning, 24
required parameter, 42–43
type mismatch, 43
e.target, 92–94
event callback objects
finished example of, 92–94
introduction to, 87–88
using, 89–92
event handlers. See also interfaces
for desktop application buttons, 410
dynamic, 89, 94–95
function definitions as, 96
for keyboard events, 246–253
for mouse events, 69, 70–73
for seeking to specific frames, 84
for sprites, 302
event listeners
addEventListner, 71, 75
in countdown clock project, 123
for dragging windows, 403
for Flipr main controls, 480–481
for Flipr score and timer displays, 493
for resizable layout of window controls, 415–417
for resizing windows, 408–410, 425
seeking to specific frames with, 83–84
simple callback functions in, 96
for splash screen timers, 479–480
for start buttons, 123
stopping playback using, 82–84
in timers, 124
for touch interactions in mobile
applications, 444
event objects
e, 73, 89–95
in mobile applications, 470
event target data example, 94–95
event types, 73, 75
Event.RESIZE event, 410
events
adding, 75
close, 404–406
desktop. See desktop events
drag, 401–403
frame. See frame events
gesture, 450–457
handlers for. See event handlers
introduction to, 68–69
listeners for. See event listeners
maximize, 398–401
minimize, 397–398
mouse. See mouse events
objects for. See event objects
resizing window, 406–410
summary of, 76
timer. See timer events
Export as Bitmap object setting, 475
external media
audio from Web, playing, 373–375
images, 370–372
introduction to working with, 369
summary of, 389
video from Web, playing, 376–378
video playback, altering, 379–382
video playback, pausing, 382–384
video playback, rewinding and seeking, 384–388

F
false value
for loop testing for, 258
introduction to, 17
NOT operator testing for, 218
AND operator testing for, 217–218
feedback, 200–201
fills, 318–320
Fireworks, 419–420, 471
FLA project files. See also Flash Professional CS5.5
classes in packages, referring to, 180–182
Document class in, creating, 169–171
Document class linking to, 302
equality and inequality in, 196–198
fonts in, 234
getName method for, 160
nested packages for, 182–183
package folders for, 176–180
publishing AIR applications and, 422
resizing windows in, 418
self-signed certificates in, 420–421
source path for, 183–186
Flash Platform. See Adobe Flash Platform
Flash Platform from Start to Finish, 423
Flash Player 10.2, defined. See also Adobe Flash Player, xvi
Flash Professional CS5.5. See also Adobe Flash Professional
Actions panel in, generally, 9–11
AIR projects in, creating, 394–396
AIR projects in, using, 424–425, 438
Android applications in, generally, 431–434
Android applications in, debugging, 458–459
Android applications in, testing, 504
Boolean variables in. See Boolean variables
Components panel in, 376
Convert to Symbol dialog box in, 5
countdown clock project in, 117–118
creating classes in, generally, 158–160
creating Document classes in, 169–171, 173
defined, xvi
drawing with ActionScript and, 302
FLA files for. See FLA project files
Format Embedding dialog box, 234
frame rate property in, 108–110
iOS applications in, creating generally, 434, 437
iOS applications in, debugging, 460
looping with drawing API in, 326–328
mobile device applications in. See mobile applications
nested packages in, 182–183
New Project dialog box in, 4
preview mode in, 470
project review in, 471–475
random number generator in, 221–224
runtime of. See Flash runtime
text fields in, 228
timeline scripting in, generally, 80
video delivery in, 376
Flash runtime. See also Flash Professional CS5.5
animation in, 333
API for artwork in, 302
default location of objects in, 16
display stack contents in, 28
displaying object locations, 16
evaluating order of operations, 61–62
introduction to, xiii, xvi
pixel hinting in, 318
text fields in, 228
timeline scripting in, generally, 80
video delivery in, 376
Flash runtime.
for loop
drawing API and, 326–328
for Flirp game board, 491
introduction to, 257–260
for puzzle solutions, 497
Format Embedding dialog box, 234
formatting text, 231–232
forwardButton instance, 83, 90
FPS (frame rate property), 108–110
frame events
ENTER_FRAME, 104–107
introduction to, 99
removing event listeners, 106–107
summary of, 111
timer events vs., 108–110
frame labels, 90
frame rate property (FPS), 108–110
frame scripts, defined, 8
frames, seeking to specific, 83–84
Froyo, 427, 505
fully qualified package names, 178
function statement, 37
functions
accepting values in, 39–43
errors when running, 42–43
fundamentals of, 36–38
introduction to, 35
for methods, 152
returning values from, 44–46
summary of, 47
G
game boards. See also Flirp game project, 491–492
game logic, 482–490
“Game Over” screen
absence of, 469
appearance of, 474
specifications for, 468
timer triggering, 494
game rules, 494–500
gameBoard, 472
GameBoard class, 482–488
gameLevelText text field, 472
gameOver method, 494
gameTimer, 472, 493–494
generic objects, 159
gesture events. See also touch interactions
introduction to working with, 450
pinch to zoom, 450–455
rotate, 455–457
GESTURE mode, 441, 450, 454
GESTURE_ROTATION event type, 455–456
garner methods
creating, 165–166
politeness of, 164
using, 166–168
GIF files, 370
Gingerbread, 504–505
glyphs, 234–235
goBackButton, 472, 481
Google, 428
gotoAndPlay() function, 84, 90
gotoAndStop() function, 84, 90–92
gotoAndStop method call, 303
graphs, 321–326
Graphic object type, 22
graphics API (application programming interface)
drawing curves with, 312–315
drawing desktop application buttons with, 396
drawing lines with, 307–312
drawing play audio buttons with, 374
drawing play video buttons with, 380
drawing shapes with, 315–318
introduction to, 301–302
looping with, 326–329
shape creation using, 303–304
initialization method
creating, 171–172
for Flipr user interfaces, 490
loops running in, 272
for splash screens, 479
for text fields, 230
instance names
in arrays, 273
assigning unique, 12, 22
on Buttons layer of Stage, 97
camel case for, 7
in classes, accessing, 154
in classes, creating, 134
in constructor parameters, adding, 146
for creating objects, 27–28
dot notation and, 15
in Flipr user interfaces, 473
of game buttons, 472
introduction to, 3–7, 19
mouse event handlers and, 70
name attribute for, 95
in objects on Stage, creating new, 28
parent property after, 470
prefixing playback commands with, 97
in timeline scripts with mouse events, 80, 95
iterations
in controlling flow of loops, 261
in for loop, 258–260
in nested loops, 263
iTunes, 436, 505

instructions screen for Flipr
contents of, 472–473
in Document class, 477
introduction to, 469
instructionsButton, 472, 481

interactions for mobile devices. See also interfaces
creating, generally, 439
dragging with touch, 442–444
gesture events for, generally, 450
long press events, 445–449
pinch to zoom gestures, 450–455
rotate gestures, 455–457
tap, 439–442
touch, 439–442
interfaces. See also events
application programming. See APIs (application programming interfaces)
defined, 67
Flipr user, 471–472
for mobile devices. See interactions for mobile devices
streamlining. See classes
INTERNET option, 439
intervals
in animation, 352–358
currentCount property for, 470
in timers, 100–102, 111
iOS platform
configuring devices for, 430
creating first application for, 434–438
debugging applications for, 460–462
testing applications for, 506–511
touch interactions on, 439
website packaging applications for, 503
IP addresses, 461
.IPA files (for Apple Devices Applications), 428, 435
iPad
introduction to, 427
on iOS platform, 428
resolution for, 434
as testing device, 429
iPhone
introduction to, 427
on iOS platform, 428
resolution for, 434
iPod touch
on iOS platform, 428
resolution for, 434
iTunes, 436, 505

icons, 419–420
idNo of clicked items, 499
if statements, 204–206
if.else if statements, 209–212
if.else statements, 206–208
Illegal assignment to variable specified as constant error, 417
images, 370–372
import flash.events.MouseEvent;, 83
import statement
for packages, 180–182
for sprites, 303
for text fields, 229
increment operators, 58–59
index element selector, 266–267
index numbers, 266–268, 491–492
inequality
demonstrating, 196–198
equality vs. See equality
testing for, 195–196
infinite loops, 259
infinitely running timers, 102, 111
init(), 171–172
_init_, 230
initialization method
creating, 171–172
for Flipr user interfaces, 490
loops running in, 272
for splash screens, 479
for text fields, 230
instances
in arrays, 273
assigning unique, 12, 22
on Buttons layer of Stage, 97
camel case for, 7
in classes, accessing, 154
in classes, creating, 134
in constructor parameters, adding, 146
for creating objects, 27–28
dot notation and, 15
in Flipr user interfaces, 473
of game buttons, 472
introduction to, 3–7, 19
mouse event handlers and, 70
name attribute for, 95
in objects on Stage, creating new, 28
parent property after, 470
prefixing playback commands with, 97
in timeline scripts with mouse events, 80, 95
iterators
in controlling flow of loops, 261
in for loop, 258–260
in nested loops, 263

H
handling scope, 87
handsets, 428
hexadecimal numbers, 231–232, 308
Honeycomb, 427
hoverButton callback function, 87–94
HTC mobile devices, 428
I
icons, 419–420
idNo of clicked items, 499
if statements, 204–206
if.else if statements, 209–212
if.else statements, 206–208
Illegal assignment to variable specified as constant error, 417
JPEG files, 370–372, 469

keeping classes polite, 164
key codes
  - character codes vs., 250–251
  - constants containing, 253
  - defined, 249
  - in keyboard shortcuts, 251–253
KeyboardEvent
  - introduction to, 246–249
  - key vs. character codes in, 250–251
  - special key recognition in, 251–253
keyboards, 227
Keychain Access, 434–435, 510–511
keyframes, 8, 85–86
kick-off meeting notes
  - countdown clock project, 116
  - on DiceOut game project, 283
  - Flipr game project, 470
kuler, 308

layers in timeline, 8
  - of buttons, 411–418
  - in desktop applications, 411–418
  - manually sketching, 240
  - of text. See layouts for text
  - of window controls in AIR, 411–418
layouts for text
  - escape sequences in, 241–245
  - introduction to, 238–240
  - Text Layout Framework, 228
left to right evaluation, 60
length property, 274
Library
  - Flipr user interfaces in, 472
  - FLVPlayback, adding, 376–378, 389
  - importing images into, 370
  - objects in. See Library objects
Library objects
  - adding dynamically, 21
  - classes of, 132, 135, 137
  - in countdown clock project, 117, 122
  - display stacks for, 27–28
  - displaying, 122
  - dynamic event handlers for, 94
  - ENTER_FRAME event and, 104
  - equality and inequality of, 197
  - Font, 234–235
  - introduction to, 5–6
  - loops creating arrays for, 269
  - on MovieClip timelines, 85
  - named assets, creating, 22–25
  - in Project setup of Flash Professional, 117
  - tween-based animation of, 80
  - values in functions of, accepting, 39–40
  - lines, drawing, 307–312
lineStyle method
  - for curves, 333–334
  - for extending Sprite class, 305
  - for gradients, 321–322
  - for lines, 307–311
  - for looping, 326–328
  - for rectangles, 317–319
  - for shapes, 304, 315–316
  - with statement and, 312
  - for strokes, 307–311
lineTo method, 310–311, 315–317
Linkage class names, 137–138
listeners. See event listeners
  - _loadImage method, 371–372
  - location
    - of curly braces, 38, 46
    - of objects, testing, 46, 196–198
    - rate of change and, 348–351
    - time and, 334–335
    - of window control buttons, 411–413, 418
    - x coordinates for. See x coordinates
    - y coordinates for. See y coordinates
  - logic operators, 216–218
  - logos, 472
  - long division with remainders, 53–54
  - long press events, 445–449
  - loops
    - animation of multiple objects with, 343–348
    - arrays and, generally, 265–266
    - break statement controlling, 261–262
    - continue statement controlling, 261–262
    - creating arrays with, 269–273
    - do loop, 264
    - in drawing with ActionScript, 326–329
    - for. See for loop
    - introduction to, 257–258
    - modifying arrays and, 266–269
    - nesting, 262–263
    - for objects already in arrays, 274–276
    - summary of, 277
Math.random method
  - in animation, 340–341
  - in drawing with ActionScript, 328
  - generating random numbers with, 221–225

Mac OS X
  - AIR applications for, 424
  - Applications folder in, 420
  - desktop applications for, 394
  - Snow Leopard, 504
  - testing mobile applications for, 506
  - main controls in Flipr, 480–482
  - manipulating objects. See also objects
    - introduction to, 3
    - parameters for, 8–18
    - summary of, 19
  - manually sketching layouts. See also
    - layouts, 240, 411–413
  - mastheads, 472
  - Math class, 221–224
  - mathematical operators
    - addition, 50–52
    - combined assignment, 55, 57
    - concatenation, 51–52
    - decrement, 58
    - division, 53–54
    - increment, 58–59
    - introduction to, 49–50
    - left to right evaluation by, 60
    - long division with remainders, 53–54
    - modulo, 53
    - multiplication, 53
    - order of operations by, evaluating, 61
    - order of operations by, forcing, 62–63
    - order of operations by, generally, 60
    - parentheses forcing order of, 62–63
    - subtraction, 50
    - summary of, 64–65
    - variables and, 55–57
  - Math.random method
    - in animation, 340–341
    - in drawing with ActionScript, 328
    - generating random numbers with, 221–225
  - matrix parameter, 324
  - maximize window control, 398–401, 411–413
  - Menu commands, 11
  - menu for Flipr, 469
  - messages
    - broadcasting mouse events, 68–69
    - buttons displaying, 70
      - “Click!” 75
    - confirmation, in array creation, 273
    - confirmation, of AIR application publication, 422
messages (continued)
  confirmation, trace statement generating, 170
debugging, on Android, 458–459
debugging, on iOS, 460
error, in named library asset creation, 24
functions generating, 44–46
greeting, functions generating, 44–45
MOUSE_OVER, 68–69
object instances broadcasting, 68
sending to Output panel, 8–11
testing application, 504
  for timeline playback, 82, 95
methods
  _clickTile, 499
  _displayInstructions, 481
  _endSplash callback, 480
  _gameOver, 494
  _loadImage, 371–372
  _positionControls, 410, 417
  _quit, 481
  _scoreGame, 498
  _setupControls, 480–481
  _setupGame, 496
private _setupGame, 496
public newGame, 496
public statement for, 152
push(), 267–268, 272
resetTimer, 494
round, 224
seekPercent, 384–385
setLabel, 382–384
setter, 164–168
setTextFormat, 233–234
setTimer, 494
splice(), 268–269
startDrag, 198, 200, 213
startMove, 403
startTouchDrag, 444, 447
stopDrag, 198, 213
stopTouchDrag, 444
for TOUCH POINT, 479
minimize window control
do for desktop applications, 397–398
location of, 418
in resizable layouts, 411
mobile applications
on Android, creating first, 431–434
configuring environment for.
  See mobile environment configuration
debugging, on Android, 458–459
debugging, on iOS, 460–462
debugging, over USB generally, 458
dragging with touch, 442–444
gesture events for, generally, 450
interactions for, generally, 439
interaction to, 427
on iOS, creating first, 434–438
long press events in, 445–449
optimizing, 463–464
pinch to zoom gestures in, 450–455
rotate gestures for, 455–457
summary of, 465
tap interactions in, 439–442
test project for, 431–438
touching for, 439–442
mobile environment configuration.
  See also mobile applications
  Android devices, setting up for testing, 504–505
  iOS devices, setting up for testing, 506–511
modulo operator
  %, 53–54
defined, 53
  for Flipr game board, 491–492
motion. See animation
Motorola mobile devices, 428–429
mouse event handlers, 70–73
mouse events
  adding to existing timeline scripts, generally, 80
callback objects, 87–94
common, 75
controlling timeline playback with, 81
drag and drop, 198–200
event target data example and, 94–95
gotoAndStop(), 92
for handling scope, 87
hovering, 74–75
in mobile applications, 442
MOUSE_OVER, 75
for MovieClips, generally, 85–86
for playback, stopping at specific frames, 83–84
for playback, stopping generally, 81–83
simple callback functions and, 96
timeline scripting with, generally, 79, 97
touch events vs., 470
MouseEvent.CLICK, 481
MouseEvent.CLICK event, 398
MouseEvent.CLICK event type, 381, 384
MOUSE_OVER events, 68–69
moveTo method, 310–311
MovieClips
  accepting values in functions in, 39–42
  Boolean variables in. See Boolean variables
class constructors for, 143–151
class statement for, 143
classes in, creating generally, 134–137
  complex conditionals for, 219
converting visual objects to, 5–7
in countdown clock project, 118
customizing button labels for, 146–147
customizing multiple properties in, 148
document class in, creating, 169–171
drag and drop for, 213
ENTER_FRAME and, 104–107
Flip user interfaces in, 471
for loop for, 259
flow of loops in, 261
frame scripts for, 8–11
game board creation in, 489
Index

new statement, 28–30, 33
new_configureWindow method, 409–410
newGame method, 494–496
nextFrame(), 92–93
Nexus One
configuring devices, 431
testing Android applications on,
504–505
as testing devices, 429
NOT operator, 218
number of intervals in timers, 100, 111
\:Number statement, 43
numbers
hexadecimal, 308
index, 266–268, 491–492
random, 215, 221–224
var statement for, 133
Order of operations
evaluating, 61
forcing, 62–63
introduction to, 60
Output panel
_getCodes method in, 250
1084: Syntax error in, 46
addition operator on, 50–52
array modification in, 268–269
assignment operator on, 55
class display screen on, 144–145
combined assignment operators in, 55
call display screen on, 144–145
concatenation operator on, 51–52
decrement operator on, 58
display stacks and, 28–29
division operator on, 53
from for loop, 260
“The function runMe was executed,” 36
getter method display in, 166, 170
if statement results on, 206
increment operator on, 58
loop code sending iterator values to, 260
looping through arrays in, 276
mathematical operators in, 50–53
mouse click events in, 72–73, 75
multiplication operator on, 53
nesting loops in, 263
AND operator in, 217
0R operator in, 218
playback, stopping, 82
random numbers in, 221
returning values from functions in,
44–46
sending messages to, 8–11
setter method display in, 166, 170
subtraction operator on, 50–52
“Tick!” displays on, 101–103
timers in, 101–103
trace statement and, 28–29, 177, 180
true value and equality statement
on, 194
x coordinate on, 12–13, 16, 200
y coordinate on, 12–13, 16, 200
p.p12 files, 434–435, 511
packages
folders for, creating, 176–180
nested, 182–183
referring to all classes in, 180–182
source paths of, 183–186

N

new statement, 28–30, 33
new_configureWindow method, 409–410
newGame method, 494–496
nextFrame(), 92–93
Nexus One
configuring devices, 431
testing Android applications on,
504–505
as testing devices, 429
NOT operator, 218
number of intervals in timers, 100, 111
\:Number statement, 43
numbers
hexadecimal, 308
index, 266–268, 491–492
random, 215, 221–224
var statement for, 133
Order of operations
evaluating, 61
forcing, 62–63
introduction to, 60
Output panel
_getCodes method in, 250
1084: Syntax error in, 46
addition operator on, 50–52
array modification in, 268–269
assignment operator on, 55
class display screen on, 144–145
combined assignment operators in, 55
call display screen on, 144–145
call display screen on, 144–145
call display screen on, 144–145
call display screen on, 144–145
call display screen on, 144–145
call display screen on, 144–145
call display screen on, 144–145
call display screen on, 144–145
call display screen on, 144–145
call display screen on, 144–145
call display screen on, 144–145
concatenation operator on, 51–52
decrement operator on, 58
display stacks and, 28–29
division operator on, 53
from for loop, 260
“The function runMe was executed,” 36
getter method display in, 166, 170
if statement results on, 206
increment operator on, 58
loop code sending iterator values to, 260
looping through arrays in, 276
mathematical operators in, 50–53
mouse click events in, 72–73, 75
multiplication operator on, 53
nesting loops in, 263
AND operator in, 217
0R operator in, 218
playback, stopping, 82
random numbers in, 221
returning values from functions in,
44–46
sending messages to, 8–11
setter method display in, 166, 170
subtraction operator on, 50–52
“Tick!” displays on, 101–103
timers in, 101–103
trace statement and, 28–29, 177, 180
true value and equality statement
on, 194
x coordinate on, 12–13, 16, 200
y coordinate on, 12–13, 16, 200
P
.p12 files, 434–435, 511
packages
folders for, creating, 176–180
nested, 182–183
referring to all classes in, 180–182
source paths of, 183–186

N
gameTimer, 472, 493–494
graphics property in, 303–304
graphics vs., 22
import statement for, 142
looping through arrays for, 274
loops creating arrays for, 269–271
in mobile applications, 470
mouse event handlers for, 69–73
naming instances in, 7
packages for. See packages
perComplete, 472–474
properties of, 17
random number generator for, 221–224
sprites for, 302–303
symbol properties for, 134–137
Tile, 498
in timeline scripting with mouse
events, 85–86
TimerExpired, 118
TimerStart, 118
without timelines, 302–303

M

MP3 files, 373–374
multi-screen projects on desktops.
See also Adobe AIR for desktop applications
multi-screen projects on mobile
devices. See Adobe AIR for mobile
applications
multiplication operator, 53
Multitouch.inputMode property, 441, 450–456

N

named Library assets. See also Library
objects, 22–25
names
fully qualified package, 178
instance. See instance names
Linkage class, 137–138
native installers, 394
native window controls
in AIR, generally, 395–398
drag, 401–403
resize, 409–410
nativeWindow object, 403
NativeWindow.displayState, 401, 407–409
nesting
condition statements, 216–217
loops, 262–263
packages, 182–183
New Project dialog box, 4

O

Object-Oriented Programming (OOP). See
also interfaces, 131, 157
objects
accessing. See access to objects
already in arrays, loops for, 274–276
animating multiple, loops for, 343–348
bitmap settings for, 475
centering, 451–452
containers for, 470
countdown clock, 117, 122
creating new, 25–28
dynamically adding. See dynamically
adding objects to Stage
event, 73, 89–95, 470
event callback, 87–94
generic, 159
Graphic object type, 22
groups of. See arrays
manipulating. See manipulating objects
moving with timers, 336–338
nativeWindow, 403
in new project creation, 4
parameters for, 12–18
sending messages to Output panel, 8–11
on Stage, 5–7
testing location of, 46, 196–198
width property of, 413
OOP (Object-Oriented Programming). See
also interfaces, 131, 157
optimization of mobile applications,
463–464
OR operator, 218
order of operations
evaluating, 61
forcing, 62–63
introduction to, 60

padding around buttons, 413
parameters
  of class constructors, 146–151
  of objects, 12–18
  of timer events, 100
parent property, 470, 499
parentheses
  in event listeners, 71
  forcing order of operations, 62–63
  location of, 46
passwords
  for AIR applications, 421
  AIR for Android, 432–433
  AIR for iOS, 435
  for keychain access, 511
pausing video, 382–384
pen, virtual, 310–315
perComplete, 472–474
phones. See also iPhone, 429–430,
  504–505
pinch to zoom gestures, 450–455
pipes, 218
pixel hinting, 318
play method of FLVPlayback, 381–382, 389
playback
  controlling, generally, 81
  FLVPlayback. See FLVPlayback
  messages, 95
  stopping, 81–83
  stopping at specific frames, 83–84
playButton, 85–93
player controls, 494–500
Player drop-down list, 394–395
PNG files, 370
polite classes, 164
_positionControls method, 410, 417
prevState(), 92–94
preview mode, 470
private
  best practices for, 163
  classes, 160, 173
  introduction to, 161–162
  restricting access with, 161–162
private _setupGame method, 496
private variables, 192–199
progressive video, 376
ProLoader class, 370–371, 389
properties
  accessing values of, 164
  align, 406
  alpha. See alpha property
  assigning to instances, 30
  cacheAsBitmap, 463
clickedTile, 500
currentCount, 470
customizing multiple, 148–149
defaultTextFormat, 233
displayState, 401
frame rate, 108–110
graphics. See graphics property
length, 274
of MovieClips, 14–19
Multitouch. inputMode, 441, 450–456
nativeWindow, 397–398
parent, 470, 499
scaleMode, 406–409
scaleX and scaleY, 447
StageAlign, 411
stageWidth and stageHeight, 328, 349
symbol, 135–136, 138
target, 89–92
text, 231–237
values of, 17, 217–218, 258
width of objects, 413
Properties panel
  in AIR, 395
  Display section of, 463
  frame rate property (FPS) in, 108–110
  introduction to, 12–14
Provisioning portal of iOS, 507–510
public
  best practices for, 164
  classes, 143
  for classes, 158
  introduction to, 158–160
  methods, 152
public newGame method, 496
Publish preview, 11
Publish Settings window, 185
publishing AIR applications, 395, 422
push() method, 267–268, 272
quit method, 481
quiz project
  escape sequences for responses in,
  241–245
  keyboardEvent handler in, 246
  questions, creating text fields for,
  228–230
  questions, properties of text for,
  231–237
  responses in layout of, 238–240
  text fields for answers in, 242–245
quote marks
  in conditionals, 213
  indicating text, 10, 19
  in private variables, 193
R
radial gradients, 326
random animation, 338–342
random number generator, 340–341, 349
random numbers, 215, 221–224
rate of change, 348–351, 355–358
"ready to run," 505
rectangles, 317
remote debugging, 458, 460–461
_remoInstructions, 481
repeating actions. See loops
reposition() function, 39–43
required parameter error, 42–43
resetTimer method, 494
resizable layouts of window controls,
  411–418
resize window controls, 406–410
resuming video playback, 384
return statement, 44–46
returning values from functions, 44–46
rewinding video, 384–388
RIM platform, 425
rise over run, 338
rotate gestures, 455–457
rotating gradient boxes, 325–326
round method, 234
rounded rectangles, 317–318
S
Safari, 506
Samsung mobile devices, 428–429
scaleMode property, 406–409
scaleX and scaleY properties, 365, 447
scaling size of visual elements, 361–365
scope of code, 87–88
score displays, 493–494, 498
_scoreGame method, 498
_scoreGame method, 498
seeking to specific frames, 83–84
seeking video, 384–388
seekPercent method, 384–385
self-signed certificates, 420–423, 432
semicolumns, 15, 46
setLabel method, 382–384
setter methods
  classes, 164
  creating, 165–166
  using, 166–168
setTextFormat method, 233–234
setTimer method, 494
Settings Window in AIR
configuring projects in, 395–396
Create Self-Signed Digital Certificate window in, 421
Icons tab in, 419–420
introduction to, 419
for iOS, 461
self-signed certificates and, 432
_setupControls method, 480–481
@sfdesigner, 501
shapes, 303–304, 315–317
ShockWave Flash (SWF), 370–372, 389
shortcuts on keyboards, 251–253
simple callback functions, 96
single quotes, 96
size of images, 370
sketching layouts manually, 240, 411–413
slope of animation
rise over run in, 338
stageWidth and stageHeight in, 349
summary of, 366
solutions and walkthroughs
ActionScript setup for countdown clock, 119–121
auto-generated imports for countdown clock, 121
contents before constructor in DiceOut, 288–290
for countdown clock, generally, 117
display objects of countdown clock, 122–123
Document class in DiceOut, generally, 284–288
Document class in Flipr, 475–478
ending game in DiceOut, 296
event listeners in DiceOut, 290–292
Flash project review in Flipr, 471–475
for Flirp game project, generally, 471
game board in Flirp, 491–492
game logic in Flirp, 482–490
game rules in Flirp, 494–500
main controls in Flirp, 480–482
player controls in Flirp, 494–500
Project setup for countdown clock, 117–118
rollIdie method in DiceOut Document class, 392–395
score displays in Flipr, 493–494
scores in DiceOut Document class, 296
splash screen in Flipr, 479
start button event listeners in countdown clock, 123
starting timer in countdown clock, 124–125
timer and timer event listeners in countdown clock, 124
timer displays in Flipr, 493–494
timer events in countdown clock, 125–126
Sound class, 374–375, 389
speaker-listener relationships. See also event listeners, 68
special key recognition, 251–253
specifications for countdown clock project, 114
for DiceOut game project, 280
for Flirp game project, 468
speed of startup, 370
splash screens
displaying, 479
in Document class, 476
game logos on, 472
introduction to, 469, 471
splice() method, 268–269
Sprite class, 302–307
sprites
class of, generally, 302–303
defined, 302
extending classes of, 305–307
in Flirp user interfaces, 471–472
stacks in Library. See also display stacks;
Library objects, 23
Stage
accepting values in functions on, 39–40
accessing objects on, 5–7
animation on. See animation comments on, creating, 31–32
countdown clock on, 122–123
display objects on, 122
display properties of mobile applications on, 463
display stacks, assigning properties to instances, 30
display stacks, creating new objects for, 25–28
display stacks, defined, 25
display stacks, working with, 29–30
dynamic event handlers for objects on, 85
dynamically adding objects to. See dynamically adding objects to Stage
editing-in-place on, 85
mouse event handlers on, 70
named library assets in, 22–25
parameters of objects on, 12–18
playButton instance on, 85–86
scaleMode property of, 406–409
spaces, working with, 32
summary of, 31
tween-based animation on, 80
StageAlign property, 411
StageScaleMode, 406–409
stageWidth and stageHeight properties, 328, 349
starfield project. See animation
startDrag method
Boolean variables and, 198, 200
summary of, 213
startGameButton, 472–474, 482
startMove method, 403
startTouchDrag method, 444, 447
startup times, 370
stop() in timeline playback, 82
in timeline scripting, 97
in timer intervals, 111
for timers, 102–103
stopButton instance, 81–82, 90–93
stopDrag method, 198, 213
stops, gradient, 321
stopTouchDrag method, 444
Streak, 429, 505
streaming video, 376, 388
string concatenation, 44–45, 51–52
strings
constructor parameters optional with, 150
customizing multiple properties with, 148
defined, 10
defining variables for, 164
getter method for, 165–167
do lines, 310–311
in method creation, 152
private, 161, 163
setter method for, 165–167
text as, 10
text field labels with, defining, 146
var statement for, 133
variables for, defining, 159
strokes, 307–312
Stucki, Ben, 464
style of lines, 307–308
style of text
changing later, 233–234
customizing, generally, 231–232
fonts, creating custom, 234–237
subtraction operator, 50
SWF (ShockWave Flash), 370–372, 389
Symantec, 422
Symbol properties window, 135–136, 138
syntax errors, 46
System Chrome
custom vs., 425
dragging windows with, 401
introduction to, 395–396
t
operator, 53
tables, 428–429
tap interactions. See also touch interactions, 439–442
target property, 89–92
testing
Android applications, 504–505
Boolean conditional, 258
conditions. See testing conditions
drag and drop, 218
for equality, 194
for false value, 217–218, 258
for loop for, 258
for inequality, 195–196
iOS applications, 506–511
location of objects, 46, 196–198
mobile applications, 431–438, 465
mobile devices for, 429–430
NOT operator for, 218
AND operator for, 217–220
OR operator for, 218
trace statement for, 202
for true value, 217–218, 258
x coordinate locations. See x coordinates
ty coordinate locations. See y coordinates
testing conditions. See also Boolean logic, advanced
generally, 202–204
if statements, 204–206
if.else if statements, 209–212
if.else statements, 206–208
loops for, 258
text
in button labels, 146
fields of, 228–230, 242–245
KeyboardEvent handler in TextFields, 246–253
layout of, escape sequences in, 241–245
layout of, generally, 238–240
style of, changing later, 233–234
style of, creating custom fonts, 234–237
style of, customizing generally, 231–232
summary of, 231–232
working with, generally, 227
Text Layout Framework (TLF), 228
TextFormat class, 231–232
thermometers, 469, 474
ticking
eample of, 101–102
generally, 100, 111
stopping timers and, 102–103
TileMovieClip, 498
tilePuzzleArray, 496–497
tiles
arrays for values of, 489
grid for, 469, 490–491
specifications for, 468
tileVariety variable, 496–497
time-based animation
introduction to, 334
summary of, 366
TimerEvent. TIMER for, 336–338
time + location, 334–335
timeline
accepting values in functions on, 39
animating thermometers with, 474
class instances in, 144
constructor parameters optional on, 150
ENTER_FRAME event in, 104
Flip user interfaces on, 472
frame scripts on, 8–10
functions on, 36
introduction to, 5
layers of, 8
mathematical operators in, 50, 53
mouse event handlers in, 71
in multiple property customizing, 148–149
new scripts, 25–26
playback, 82
scripting. See timeline scripting with mouse events
Timeline panel, 8, 26
timeline scripting with mouse events
event callback objects in, finished
eample of, 92–94
event callback objects in, generally, 87–88
event callback objects in, using, 89–92
event target data example, 94–95
gotoAndStop() shortcuts for, 92
handling scope in, 87
introduction to, 79
in MovieClips, 85–86
playback, controlling generally, 81
playback, stopping, 81–83
playback, stopping at specific frames, 83–84
project for (ball moving across screen), introducing, 80
simple callback functions in, 96
summary of, 97
Timer, defined, 100
TimerEvent, 493
timer events
callback functions for, 125–126
frame events vs., 108–110
introduction to, 99
parameters of, 100
stopping, 102–103
summary of, 111
using, 100–102
TIMER COMPLETE event, 100–102
TimerEvent, 100
TimerEvent. TIMER, 336–338, 445
timers
_timerSecond for game displays, 493
in animation, 336–338
displaying, 493–494
event listeners for, 134
events in. See timer events
“Game Over” screen triggered by, 494
gameTimer, 493–494
intervals in, 355–358
in mobile applications, 470
resetTimer method for, 494
setTimer method for, 494
for splash screens, 479–480
starting with callback functions, 124–125
ticking of, 100–103, 111
_TIMER event for game displays, 493
_timerSecond, 493
Times New Roman, 231
times, startup, 370
TLF (Text Layout Framework), 228
toggle buttons, 218, 382–384
totalRight variable, 496, 498–499
touch interactions. See also gesture events
creating, 439–442
mouse events vs., 470
splash screens and, 479
TOUCH_BEGIN event type, 443–448
 TOUCH_END event type, 443–449
TouchEvent .TOUCH_TAP, 481
TouchEvent Document class, 450
TOUCH_INPUT interaction type, 442
TOUCH_POINT method, 441, 479
touchPointID, 444
trace statement
arrays and, 266
calling functions with, 44–47
dev in iPadOS with, 462
defined, 10–12
in timeline scripting, 89
TransformGestureEvent, 450–456
transparency, 308, 323
true value
for loop testing for, 258
true value
for class variables, generally, 133
in for loop, 259
for mathematical operators, 56
text fields, creating with, 228, 230
variables
Boolean. See Boolean variables
of classes, generally, 133
defining, 159
introduction to, 12
of mathematical operators, 55–57
private, 192–199
specified as constant, 417
tileVariety, 496–497
totRight, 496, 498–499
var statement for, see var statement
Vectors, 463
Verdana font, 234
VeriSign, 422
video from Web, 376–378
video playback
altering generally, 379–382
pausing, 382–384
rewinding, 384–388
seeking, 384–388
VideoButton class, 382–384
virtual pen, 310–315
visibility settings, 18
visual design review
of countdown clock project, 115
of DiceOut game project, 281–282
of Flipp game project, 469
with statement, 311–312, 328
workspace for ActionScript development, 136–137

X
x coordinates
in adding animation elements, 358–361
in animation brightness, 352–355
in animation fine-tuning, generally, 349
in animation fluidity and rate of change, 355–358
curveTo method and, 312–315
feedback identifying, 200–201
for Flipp game board, 491
if statement testing, 204–206
if.else if statement testing, 209–212
if.else statement testing, 206–208
introduction to, 12–15
in loops animating multiple objects, 343–348
AND operators testing, 219–220
in random animation, 339–342
in scaling size of animation elements, 361–365
testing for, 196–200, 202–203
window control buttons on, 412–413

Y
y coordinates
in adding animation elements, 358–361
in animation brightness, 352–355
in animation fine-tuning, generally, 349
in animation fluidity and rate of change, 355–358
curveTo method and, 312–315
feedback identifying, 200–201
for Flipp game board, 492
if statement testing, 204–206
if.else if statement testing, 209–212
if.else statement testing, 206–208
introduction to, 12–15
in loops animating multiple objects, 343–348
AND operators testing, 219–220
in random animation, 339–342
in scaling size of animation elements, 361–365
testing for, 196–200, 202–203
window control buttons on, 412–413