Preface

Welcome to *ActionScript for Flash MX: The Definitive Guide,* Second Edition! This edition sports massive changes from the first edition, with hundreds of pages of new material and exhaustive rewrites that bring old material up to date with best practices for Flash MX. I hope you're as excited to read it as I was to write it!

Like the first edition, this book teaches ActionScript from the ground up, covering both basic concepts and advanced usage, but with a special focus on Macromedia Flash MX techniques. In Part I, we'll explore ActionScript fundamentals—from variables and movie clip control to advanced topics such as objects, classes, and server communication. In Part II, the *Language Reference*, we'll cover every object, class, property, method, and event handler in the core ActionScript language. You'll use the *Language Reference* regularly to learn new things and remind yourself of the things you always forget, so keep this book on your desk, not on your shelf!

Though ActionScript's complexity has increased in Flash MX, you do not have to be a programmer to read this book. I have continued to be mindful of the beginner throughout this edition. The text moves pretty quickly, but a prior knowledge of programming is not required to read it. All you need is experience with the non-ActionScript aspects of Flash and an eagerness to learn. Of course, if you are already a programmer, so much the better; you'll be applying your code-junkie skills to ActionScript in no time. To make the transition to Flash easier for experienced programmers, I've made a special effort to draw helpful analogies to languages such as JavaScript, Java, and C.

Above all, this book truly is a Definitive Guide to ActionScript in Flash MX. It's the product of nearly four years of research, thousands of emails to Macromedia employees, and feedback from users of all levels. I hope that it is self-evident that I've suffused the book with both my intense passion for the subject and the painfully won, real-world experience from which you can benefit immediately. It covers ActionScript with exhaustive authority and—thanks to a technical review by Gary Grossman, the creator of ActionScript—with unparalleled accuracy.

Second Edition Quick Start

If you're a returning first-edition reader dying to sink your teeth into this edition, here are the highlights I recommend you start with. But don't end your exploration with this list. Read on to learn about many more important updates to this edition.

The following chapters in Part I, *ActionScript Fundamentals*, have been heavily rewritten and enhanced. They cover some of the most exciting additions, such as components, and meaningful changes to the way ActionScript handles events and deals with objects.

- Chapter 9, *Functions*
- Chapter 10, *Events and Event Handling*
- Chapter 12, *Objects and Classes*
- Chapter 14, *Movie Clip Subclasses and Components*

See also the revised and new appendixes, especially:

- Appendix C, *Backward Compatibility and Player Build Updates*
- Appendix E, *HTML Support in Text Fields*
- Appendix F, *Support for GET and POST*
- Appendix G, *Flash UI Component Summary*
- Appendix H, *Embedding a Flash Movie in a Web Page*

The following entries in Part II, the *Language Reference*, are either all-new or have been heavily revised since the first edition. For example, you'll want to read up on the new *SharedObject* object and check out the Drawing API methods added to the *MovieClip* class.

- *Accessibility* object
- *Button* class
- *Capabilities* object
- *Function* class
- `_global` object
- `#initclip` and `#endinitclip` pragmas
- *LoadVars* class
- *LocalConnection* class
- *MovieClip* class (new events and the Drawing API)
- *Object* class
- *setInterval( )* and *clearInterval( )* global functions
- *SharedObject* object
- *Sound* class

- *Stage* object
- *System* object
- *TextField* class
- *TextFormat* class
- Listener Events for *Key*, *Mouse*, *TextField*, and *Stage* (see Table P-1)

What's New in Flash MX ActionScript

ActionScript evolved tremendously from Flash 5 to Flash MX (as the authoring tool is known) and the corresponding Flash Player 6, and this book has evolved along with it. See Table P-2 in this Preface for details on the Flash version naming conventions.

To preview many of the new features in action, visit:
http://www.moock.org/webdesign/lectures/newInMX

Table P-1 provides a high-level overview of the major additions to ActionScript and tells you where to find more information about each new topic in this book. Unless otherwise stated, cross-references are to Part II, the *Language Reference*.

Table P-1. New features in Flash MX ActionScript

Feature	For details, see...
Drawing API: draw strokes, shapes, and fills at runtime using new *MovieClip* methods	*MovieClip.beginFill( )*, *MovieClip.beginGradientFill( )*, *MovieClip.clear( )*, *MovieClip.curveTo( )*, *MovieClip.endFill( )*, *MovieClip.lineStyle( )*, *MovieClip.lineTo( )*, *MovieClip.moveTo( )*; "Drawing in a Movie Clip at Runtime" in Chapter 13
Load JPEG-format images at runtime	*MovieClip.loadMovie( )*, *loadMovie( )*
Load MP3-format sounds at runtime	*Sound.loadSound( )*
Check the length of a sound and the amount of time it has been playing	*Sound*`.position`, *Sound*`.duration`
Detect when a sound finishes playing	*Sound.onSoundComplete( )*
Create, manipulate, and format text fields at runtime	The *TextField* class, the *TextFormat* class, *MovieClip.createTextField( )*
Mask or unmask a movie clip at runtime	*MovieClip.setMask( )*
Create movie clips from scratch at runtime	*MovieClip.createEmptyMovieClip( )*
Determine a movie clip's depth at runtime	*MovieClip.getDepth( )*
Execute a function or method periodically	*setInterval( )*, *clearInterval( )*
Manipulate XML, string, and array data faster due to Flash Player performance improvements	The *XML* class, the *String* class, the *Array* class
Store data locally (much like JavaScript cookies)	The *SharedObject* object

Table P-1. New features in Flash MX ActionScript (continued)

Feature	For details, see...
Create packaged code modules with *MovieClip* subclasses and components	`#initclip`, `#endinitclip`, *Object.registerClass( )*, *attachMovie( )*; Chapter 14
Communicate between two Flash Players on the same computer	The *LocalConnection* class
Declare global variables	`_global`; "Movie Clip Variables and Global Variables" in Chapter 2
Use international characters in the Unicode character set	"The String Type" in Chapter 4, Appendix C
Define event handlers on movie clips using callback functions	Chapter 10
Use event listeners to respond to events from any object	Chapter 10 and *Key.addListener( )*, *Mouse.addListener( )*, *Stage.addListener( )*, *Selection.addListener( )*, *TextField.addListener( )*
Add button behavior to a movie clip	"Using Movie Clips as Buttons" in Chapter 13
Control button objects at runtime	The *Button* class
Make content accessible to screen readers for the visually impaired	The *Accessibility* object
Check the movie width and height at runtime, and reposition movie elements when the movie is resized	`Stage.height`, `Stage.width`, *Stage.onResize( )*
Use lexical and nested function scope, or execute a function as a method of an arbitrary object	*Function.call( )*, *Function.apply( )*; "The Scope Chain" in Chapter 2; "Function Scope" in Chapter 9
Access Player and system information such as screen resolution, operating system, and current language	The *Capabilities* object
Capture keyboard and mouse input events with a centralized input API	The *Key* object, the *Mouse* object
Load variables using an intuitive variable loading class rather than the *loadVariables( )* function	The *LoadVars* class
Monitor the download progress of XML or loading variables	*XML.getBytesLoaded( )*, *LoadVars.getBytesLoaded( )*
Control the tab order for buttons, text fields, and movie clips	*`TextField`*`.tabIndex`, *`Button`*`.tabIndex`, *`MovieClip`*`.tabIndex`
Turn off the hand cursor for buttons	*`Button`*`.useHandCursor`, *`MovieClip`*`.useHandCursor`
Add getter/setter properties to an object, and receive notification when a property changes	*Object.addProperty( )*, *Object.watch( )*

What's New in the Second Edition

The second edition of *ActionScript for Flash MX: The Definitive Guide* is not merely a "tack-on" update to the first edition (which was titled *ActionScript: The Definitive Guide*). The entire text has been revised and restructured to highlight the latest Flash MX ActionScript features. Nearly every paragraph has been updated, and 400 pages have been added to cover ActionScript's new capabilities. Legacy descriptions of Flash 4 ActionScript syntax have been moved from the body of the book to

Appendix C or online technotes. We made this choice to keep the book streamlined, although it is still considerably beefier than the first edition. By the time you read this, Flash Player 6 will be nearly ubiquitous, so it doesn't make sense to cover Flash 4 in detail anymore. We cover enough of it to help you understand and upgrade any legacy code you may own or encounter. We've also paid close attention to changes between Flash 5 and Flash 6 to help you understand the new paradigms and upgrade legacy code. The legacy code examples from the first edition will all remain available at *http://www.moock.org/asdg/codedepot*.

Updated Code Examples

All code examples from the first edition have been rewritten to use Flash MX syntax and best practices. For example:

- The quiz samples now use callback functions—rather than Flash 5–style *on()* handlers—for button event handlers.
- Text fields that were formerly drawn in the authoring tool are now generated programmatically with *createTextField()*.
- Classes are defined on `_global` (the new property that holds global variables)
- The object-oriented *LoadVars* class is used instead of the older *loadVariables()* global function.

Likewise, dozens of new Flash MX-specific examples have been added. Here are just a few of the interesting ones:

- A completely code-based, object-oriented quiz, downloadable from the online Code Depot (described later in this Preface)
- A configurable text ticker (see *`TextField`*`.hscroll`)
- An array-to-table converter (see *`TextFormat`*`.tabStops`)
- A sound preloader (see *Sound.getBytesLoaded()*)

Hundreds of Tweaks

Subtle details have been added throughout this book to augment the first edition's content. Here are just a few of the hundreds of tweaks made:

- *`MovieClip`*`._x` discusses *twips* (the minimum distance a clip can be moved).
- *`MovieClip`*`._visible` warns that button events don't fire when `_visible` is `false`.
- *XML.parseXML()* covers CDATA and predefined XML entities (&, <, >, ", and ') at length.
- *MovieClip.getBytesLoaded()* features a list of possible return values based on the asynchronous execution of *loadMovie()*.

- Chapter 2 discusses qualified and unqualified variable references and *Hungarian notation*.
- Chapter 4 explicitly contrasts `null` with *delete* and `undefined`.

Of course, there are plenty of not-so-subtle changes too. We'll look at them next.

Major Revisions Since the First Edition

The following list describes the major content and structural changes in this second edition. Note that some of these chapters were in Part II, *Applied ActionScript*, in the first edition. Other material from the first edition's Part II was redistributed elsewhere in this second edition, and some content was moved to online technotes. Despite the organizational change, rest assured that this second edition includes dozens of applied examples sprinkled liberally throughout the entire book. The *Language Reference*, formerly Part III in the first edition, is now Part II.

Chapter 1, *A Gentle Introduction for Nonprogrammers*

- Added an introduction to object-oriented programming
- Revised the quiz tutorial for Flash MX
- Revised the event handler section for Flash MX

Chapter 2, *Variables*

- Added recommended suffixes for variable names
- Added global variable coverage
- Added a section on loading external variables
- Added an explicit discussion of the *scope chain*

Chapter 3, *Data and Datatypes*

- Added the section "Copying, Comparing, and Passing Data" (formerly in Chapter 15)

Chapter 4, *Primitive Datatypes*

- Added coverage of Unicode

Chapter 5, *Operators*

- Added coverage of the strict equality and *instanceof* operators

Chapter 6, *Statements*

- Added *switch* statement coverage
- Revised the description of *with* to include the scope chain
- Removed the legacy *call* statement (now covered in the *Language Reference* only)

Chapter 8, *Loop Statements*

- Added a section on using *setInterval()* to execute code repeatedly
- Revised "Timeline and Clip Event Loops" to use Flash MX features (*MovieClip.createEmptyMovieClip()* and the *MovieClip.onEnterFrame()* handler)

Chapter 9, *Functions*

- Added a section on the differences between function literals and the *function* statement
- Added coverage of nested functions
- Revised "Function Scope" to cover *lexical scope* in more detail
- Revised the quiz tutorial for Flash MX

Chapter 10, *Events and Event Handling*

- Added complete coverage of event handler properties
- Added coverage of event listeners, new in Flash MX
- Added an in-depth discussion of scope, including Table 10-1, which compares old scope rules to new scope rules
- Added a description of the `this` keyword within various handlers, including a summary in Table 10-2
- Moved all specific button and movie clip event descriptions to the *Language Reference* (see also Table 10-3)

Chapter 11, *Arrays*

- Added coverage of the *Array.sortOn()* method
- Revised the quiz tutorial for Flash MX

Chapter 12, *Objects and Classes*

- Revised the chapter entirely to focus more squarely on the process of making a class with methods and properties
- Added coverage of Flash MX's *super* keyword, used to invoke a superclass constructor and its methods
- Added a formal discussion of the *prototype chain*
- Added a formal discussion of issues with standard *superclass assignment*
- Added a section on static methods and properties
- Added a description of rendering an object to screen
- Added an object-oriented programming (OOP) application template
- Added an "OOP Quick Reference" section
- Added a brief discussion of UML and design patterns

Chapter 13, *Movie Clips*

- Added information on creating a blank movie clip from scratch using *MovieClip.createEmptyMovieClip( )*
- Added a section on drawing in a movie clip at runtime using the new Drawing API
- Added a section on implementing button behavior for a movie clip
- Added a section on handling input focus for movie clips
- Revised (fixed) the first edition's partially erroneous description of *MovieClip.duplicateMovieClip( )* depths
- Moved the list of *MovieClip* methods and properties to the *Language Reference*
- Moved the legacy *Tell Target* discussion to Appendix C
- Updated the clock example to use Flash MX best practices
- Removed the quiz example, which is superceded by the new downloadable OOP quiz (the legacy version is still available online)

Chapter 14, *Movie Clip Subclasses and Components* (all new)

- Covers how to make movie clip subclasses (specialized types of movie clip symbols associated with a class)
- Covers how to create a basic component, of which the Flash UI Components are a complex example

Chapter 15, *Lexical Structure* (previously Chapter 14)

- Revised the list of reserved words
- Removed and redistributed old Chapter 15, content as follows:
  - Moved "Copying, Comparing, and Passing Data" to Chapter 3
  - Moved "Bitwise Programming" to online technote at *http://www.moock.org/asdg/technotes*
  - Removed "Advanced Function Scope Issues" (the issue discussed was fixed in Flash MX)
  - Moved "The MovieClip Datatype" to online technote at *http://www.moock.org/asdg/technotes*

Chapter 16, *ActionScript Authoring Environment*

- Revised the section on legacy Smart Clips to cover new Flash MX Components architecture instead

Chapter 17, *Building a Flash Form*

- Revised the code example and tutorial to use *LoadVars* class instead of *loadVariables( )*

Redistributed old Chapter 18, *On-Screen Text Fields* (in first edition only)

- Contents of the entire chapter moved to the *Language Reference* (under *TextField* class) and to Appendix E (and augmented with substantial additions to the *TextField* class)

Removed old Chapter 19, *Debugging* (in first edition only)

- Entire chapter moved to online technote at *http://www.moock.org/asdg/technotes*

Part II, *Language Reference* (formerly Part III)

- Earlier in this Preface, we highlighted the major changes and additions to the *Language Reference*. For a complete list of new methods, properties, classes, objects, global functions, and directives added to the *Language Reference*, see *http://www.moock.org/webdesign/lectures/newInMX*. (Note that *CustomActions* and *LivePreview* are not included in the *Language Reference*, as discussed next.)

What's Not in This Book

Although this book is vast, ActionScript is vaster. It is no longer feasible to cover every possible ActionScript topic within the confines of a single book. We made a conscious editorial decision in this edition to omit formal coverage of the following items (though these topics are covered in passing where relevant):

- Features used exclusively to extend the Flash MX authoring tool (e.g., *CustomActions* and *LivePreview*). These topics are covered in Macromedia's online article "Creating Components in Flash MX" at *http://www.macromedia.com/support/flash/applications/creating_comps*.
- Macromedia's library of Flash UI Components, which extend the authoring tool beyond the core language. See Appendix G, *Flash UI Component Summary*, for a summary of Flash UI Components properties and methods. For resources that cover Flash UI Components in depth, see "Summary" in Chapter 14.
- The Macromedia Flash Communication Server MX (Comm Server) API (e.g., *Remote SharedObject*, *Camera*, *Microphone*, *NetConnection*, and *NetStream*). Comm Server is used to create multiuser web applications with audio and video. See *http://www.macromedia.com/software/flashcom/* for details.
- The basics of the Flash MX authoring tool. However, if you are a programmer who is new to Flash, we give you enough hints so you can input the code examples and follow along. To learn Flash MX animation and graphic design, start with the online help and manual; then explore the web sites listed at *http://www.moock.org/moockmarks*.

There is no CD in the back of the book, but all the code examples can be downloaded from the online Code Depot (cited later in this Preface).

Undocumented ActionScript Features

The Flash development community has a knack for unearthing so-called *undocumented features* of ActionScript—internal abilities of the language that are not officially released or sanctioned for use by Macromedia. In general, use of undocumented features is not recommended because:

- They are not tested for external use and may therefore contain bugs or be unstable.
- They may be removed from future versions of the language without warning.

In this book, we chose to focus on providing the best possible documentation for features that are supported but which may be poorly documented or misdocumented. Therefore, wholly undocumented or unsupported features are not covered unless:

- Macromedia sources have supplied or confirmed the information directly; or
- Use of the feature is so widespread that it demands discussion.

In either case, descriptions in this book of undocumented features include the appropriate warning label in full view. This book covers the following undocumented features:

- `__proto__` (as used to establish inheritance)
- *ASBroadcaster* (partial coverage only, in Chapter 12)
- *ASSetPropFlags()* (partial coverage only, in Chapter 8)
- *LoadVars.decode()*
- *LoadVars.onData()*
- *Object.hasOwnProperty()*
- *System.showSettings()*
- *`TextField`*`.condenseWhite`
- *`TextFormat`*`.font`'s multiple font abilities
- The *XMLNode* class

To see what the ActionScript sleuths have discovered, visit (with prudence):

http://chattyfig.figleaf.com/flashcoders-wiki/index.php?Undocumented%20Features

Flash Naming Conventions

With the introduction of the MX family of products, including Flash MX, Macromedia abandoned a standard numeric versioning system for its Flash authoring tool. The Flash Player, however, is still versioned numerically. Table P-2 describes the naming conventions used in this book for Flash versions.

Table P-2. Flash naming conventions used in this book

Name	Meaning
Flash MX	The Flash MX authoring tool (as opposed to the Flash Player)
Flash Player 6	The Flash Player, version 6. The Flash Player is a browser plugin for major web browsers on Windows and Macintosh. There are both ActiveX and Netscape-style versions of the plugin, but they are referred to collectively as "Flash Player 6" except where noted, such as under *Accessibility* in the *Language Reference*.
Flash Player *x*.0.*y*.0	The Flash Player, specifically, the release specified by *x* and *y*, as in Flash Player 6.0.47.0. See `capabilities.version` in the *Language Reference* for details.
Flash 6	Short for "Flash Player 6," used primarily in the *Language Reference* or wherever the distinction between Flash MX (the authoring tool) and Flash Player 6 (the browser plugin) is irrelevant.
Flash 5 authoring tool	The Flash 5 authoring tool (as opposed to the Flash Player), which came before Flash MX
Flash Player 5	The Flash Player, version 5
Flash 5	Short for "Flash Player 5," used primarily in the *Language Reference* or wherever the distinction between Flash 5 (the authoring tool) and Flash Player 5 (the browser plugin) is irrelevant.
Flash 2, Flash 3, and Flash 4	Versions of the Flash Player prior to version 5, used primarily in the *Language Reference* to indicate which versions of Flash support the given feature.
Standalone Player	A version of the Flash Player that runs directly off the local system, rather than as a web browser plugin or ActiveX control.
Projector	A self-sufficient executable that includes both a *.swf* file and a Standalone Player. Projectors can be built for either the Macintosh or Windows operating system using Flash's File → Publish feature.

What Can ActionScript Do?

ActionScript is used to create all kinds of interactive applications, typically for web-based use. Here are just a few possibilities: an MP3 player, a multiuser drawing application, a 3D walkthrough of a home, an online store, a message board, an HTML editor, and the game Pac-Man. Each of these applications uses a combination of ActionScript's capabilities, a sampling of which follows. Begin thinking about how you can combine these techniques to build your applications.

Timeline Control

Flash movies are composed of frames residing in a linear sequence called the *timeline*. Using ActionScript, we can control the playback of a movie's timeline, play segments of a movie, display a particular frame, halt a movie's playback, loop animations, and synchronize animated content. *Movie clips* within a main movie each have their own timeline.

Interactivity

Flash movies can accept and respond to user input. Using ActionScript, we can create interactive elements such as:

- Buttons that react to mouseclicks (e.g., a classic navigation button)
- GUI elements such as list boxes, combo boxes (a.k.a. drop-down menus), and check boxes
- Content that animates based on mouse movements (e.g., a mouse trailer)
- Objects that can be moved via the mouse or keyboard (e.g., a car in a driving game)
- Text fields that display information on screen or allow users to supply input to a movie (e.g., a fill-in form)

Visual and Audio Content Control

ActionScript can be used to examine or modify the properties of the audio and visual content in a movie. For example, we can change an object's color and location, reduce a sound's volume, or set the font face of a text block. We can also modify these properties repeatedly over time to produce unique behaviors such as animated effects, physics-based motion, and collision detection.

Programmatic Content Generation

Using ActionScript, we can generate visual and audio content directly from a movie's Library or by duplicating existing content on the Stage. In Flash MX, we can use the *MovieClip* class's Drawing API, *createEmptyMovieClip()* method, and *createTextField()* method to create graphics and text from scratch at runtime. Programmatically generated content may serve as a strictly static element—such as a random visual pattern—or as an interactive element—such as a button in a dialog box, an enemy spaceship in a video game, or an option in a pull-down menu.

Server Communication

One of the most common ways to extend Flash's functionality is via communication with some server-side application or script, such as Macromedia ColdFusion MX or a Perl script. Although communicating with ColdFusion is largely the purview of Macromedia Flash Remoting MX (Flash Remoting), the core ActionScript language provides a wide variety of tools for sending information to, and receiving information from, any server-side application or script (e.g., Java, PHP, ASP, etc.). The following applications all involve server communication:

Link to a web page
See *getURL()*.

Guest book
See the *LoadVars* and *XML* classes, Chapter 17, and the Code Depot, described in the next section.

Chat application
See the *XMLSocket* class and the example at *http://www.moock.org/chat*.

Multiplayer networked game
See the *XMLSocket* class and *http://www.moock.org/unity*.

E-commerce transaction
See the *LoadVars* and *XML* classes.

Personalized site involving user registration and login
See the *LoadVars* and *XML* classes.

Detailed implementations of even this limited number of potential ActionScript applications are beyond the scope of this book. Instead, our goal is to give you the fundamental skills to explore the myriad other possibilities on your own. This is not a recipe book—it's a lesson in cooking code from scratch. What's on the menu is up to you.

The Code Depot

We'll encounter dozens of code samples over the upcoming chapters. To obtain relevant source files and many other tutorial files not included in the book, visit the online Code Depot, posted at:

http://www.moock.org/asdg/codedepot

The Code Depot is an evolving resource containing real-world ActionScript applications and code bases. Here's a selected list of samples you'll find in the Code Depot:

- A multiple-choice quiz
- A pan-and-zoom image viewer
- Text field tools, such as an array-to-table converter and a configurable text ticker
- An XML-based chat application
- A guest book application
- A custom mouse pointer and button
- An asteroids game code base
- Programmatic motion effects
- Demos of HTML text fields

- Preloaders
- String manipulation
- Interface widgets, such as slider bars and text scrollers
- Mouse trailers and other visual effects
- Volume and sound control

Additionally, any book news, updates, technotes, and errata will be posted here.

Showcase

Practically every Flash site in existence has at least a little ActionScript in it. But some sites have, shall we say, more than a little. Table P-3 presents a series of destinations that should provide inspiration for your own work. See also the sites listed in Appendix A and the author's bookmarks at *http://www.moock.org/moockmarks*.

Table P-3. ActionScript Showcase

Topic	URL
Experiments in design, interactivity, and scripting	*http://www.yugop.com* *http://www.praystation.com** *http://www.presstube.com* *http://www.pitaru.com* *http://www.flight404.com* *http://www.bzort-12.com* *http://www.benchun.net/mx3d/** *http://www.protocol7.com** *http://www.uncontrol.com** *http://flash.onego.ru** *http://www.figleaf.com/development/flash5** *http://nuthing.com* *http://www.deconcept.com* *http://www.natzke.com*
Games	*http://www.orisinal.com* *http://www.gigablast.com* *http://www.sadisticboxing.com* *http://www.huihui.de* *http://www.sarbakan.com* *http://www.electrotank.com/games/multiuser* *http://www.titoonic.dk/products/games/spider* *http://content.uselab.com/acno* *http://www.neave.com/webgames*

Table P-3. ActionScript Showcase (continued)

Topic	URL
Interface, applications, and dynamic content	*http://www.mnh.si.edu/africanvoices* *http://www.curiousmedia.com* *http://www.smallblueprinter.com* *http://davinci.figleaf.com/davinci* *http://host.oddcast.com* *http://www.enteryourinformation.com/broadmoor/onescreen.cfm*

* Downloadable *.fla* files provided. Otherwise, only *.swf* files available.

Typographical Conventions

In order to indicate the various syntactic components of ActionScript, this book uses the following conventions:

Menu options
: Menu options are shown using the → character, such as File → Open.

`Constant width`
: Indicates code samples, clip instance names, frame labels, property names, and variable names. Variable names often end with the suffixes shown in Table 2-1 (such as `_mc` for variables that refer to movie clip instances). Although using these suffixes is considered the best practice, we sometimes avoided them when we found they made the surrounding text substantially more difficult to read. For brevity, therefore, the preferred suffixes have sometimes been omitted.

Italic
: Indicates function names, method names, class names, layer names, URLs, filenames, and file suffixes such as *.swf*. In addition to being italicized, method and function names are also followed by parentheses, such as *duplicateMovieClip()*.

`Constant width bold`
: Indicates text that you must enter verbatim when following a step-by-step procedure. **`Constant width bold`** is also used within code examples for emphasis, such as to highlight an important line of code in a larger example.

`Constant width italic`
: Indicates code that you must replace with an appropriate value (e.g., *`your name here`*). *`Constant width italic`* is also used to emphasize variable, property, method, and function names referenced in comments within code examples.

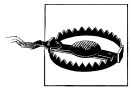
In the *Language Reference*, we played around with some font conventions. The following conventions looked the best, while maintaining consistency with our overall approach, so we went for them:

- Class-level properties are shown with both the class name and property in `constant width`, because they should both be entered verbatim, as shown (e.g., `Stage.width`, `Math.NaN`).
- Instance-level properties are shown with the class or object instance in *`constant width italic`*, because the placeholder should be replaced by a specific instance. The property itself is shown in `constant width` and should be entered as shown (e.g., *`Button`*`.tabEnabled`, where *`Button`* should be replaced with a button instance).
- Method and function names, and the class or object to which they pertain, are always shown in italics and followed by parentheses, as in *MovieClip.duplicateMovieClip()*. Refer to the *Language Reference*, surrounding material, and nearby examples to determine whether to include the class name literally, as in *TextField.getFontList()*, or replace it with an instance name, such as *ball_mc.duplicateMovieClip()*.
- Within the *Language Reference*, for brevity, we often omit the class name when discussing a property or method of the class. For example, when discussing the `htmlText` property of the *TextField* class, when we say "set the `htmlText` property," you should infer from context that we mean, "set the *`someField_txt`*`.htmlText` property, where *`someField_txt`* is the identifier for your particular text field."
- In some cases, an object property contains a reference to a method or callback handler. It wasn't always clear whether we should use `constant width` to indicate that it is a property (albeit one storing a method name) or *italics* and parentheses to indicate it is a method (albeit one stored in a property). If the line between a property referring to a method and the method itself is sometimes blurred, forgive us. To constantly harp on the technical difference would have made the text considerably less accessible and readable.
- When summarizing properties for a class, the properties may be shown in *italics*, rather than `constant width`, to save space. This applies only when the properties are summarized under a *Properties* heading and they aren't followed by parentheses, so it is clear that they're properties and not methods.

If any or all of this is confusing now, it will be clear by the time you get to the *Language Reference*, having read about objects, classes, and movie clips in Chapters 12, 13, and 14.

Pay special attention to notes and warnings set apart from the text with the following icons:

This is a tip. It contains useful information about the topic at hand, often highlighting important concepts or best practices.



This is a warning. It helps you solve and avoid annoying problems or warns you of impending doom. Ignore at your own peril.

We'd Like to Hear from You

We have tested and verified the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). Please let us know about any errors you find, as well as your suggestions for future editions, by writing to:

O'Reilly & Associates, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international/local)
(707) 829-0104 (fax)

We have a web page for the book, where we list errata, examples, or any additional information. You can access this page at:

http://www.oreilly.com/catalog/actscript2

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, software, Resource Centers, and the O'Reilly Network, see our web site at:

http://www.oreilly.com

Acknowledgments

As with the first edition, this book would be a mere shadow of itself without the incredible contributions of Macromedia Flash MX's engineering, quality assurance, support, and product management teams. In particular, I can never thank Gary Grossman enough for his critiques, guidance, and patience, not to mention writing the Foreword. Other Macromedians who helped shape this text include: Jonathan Gay, Jeremy Clark, Eric Wittman, Michael Williams, Pete Santangeli, Matt Wobensmith, Ben Chun, Troy Evans, Lee Thomason, Bentley Wolfe, John Dowdell, Rebecca Sun, Janice Pearce, Brian Dister, Henriette Cohn, Jeff Mott, Michael Morris, Deneb Meketa, Tinic Uro, Robert Tatsumi, Colm McKeon, and Mike Chambers.

This book's editor is Bruce Epstein, who I am convinced is superhuman. His knowledge of writing and programming is exceptional, and his ability to bestow that knowledge upon a text is astonishing. I am uncommonly fortunate to be coached by such an outstanding editor (and author in his own right).

Next, it is my honor to present the technical reviewers of this edition, all of whom are members of Macromedia Flash MX's engineering team: Gary Grossman, Chris Thilgen, Gilles Drieu, Nigel Pegg, Slavik Lozben, and Michael Richards. Erica Norton edited the first edition. Thank you, my friends, for your time and devotion.

The beta readers for this edition are all renowned Flash developers for whom I have immense respect: Robert Penner (*http://www.robertpenner.com*), Dave Yang (*http://www.quantumwave.com*), Branden Hall (*http://www.waxpraxis.org*), Amit Pitaru (*http://www.pitaru.com*), Michael Kay (*http://www.peep.org/wizard/*), and Veronique Brossier (*http://www.v-ro.com*). This book's accuracy is in many cases the result of their keen eyes.

Thanks to Tim O'Reilly for setting a standard of thoroughness, quality, and accuracy in everything he publishes. And thanks to O'Reilly's Brian Sawyer, Claire Cloutier, Glenn Bisignani, Mike Sierra, Rob Romano, Edie Freedman, Sandy Torre, and the many copyeditors, indexers, proofreaders, and sales and marketing folks at O'Reilly who helped bring this book to the shelves.

I owe recognition to my good friend Derek Clayton for regularly sharing his programming expertise with me. Derek contributed the Perl code in Chapter 17, the Java *XMLSocket* server in the *Language Reference*, and a generic flat file database system, all available from the online Code Depot. He is also the lead developer of Unity Socket Server, moock.org's commercial application for creating multiuser applications in Flash (*http://www.moock.org/unity*).

To the Flash community: thank you for the inspiration and beauty you create. In particular, thanks to James Patterson, Yugo Nakamura, Naoki Mitsuse, Joshua Davis, James Baker, Marcell Mars, Phillip Torrone, Robert Reinhardt, Mark Fennell, Josh Ulm, Darrel Plant, Todd Purgason, John Nack, Jason Krogh, Hillman Curtis, Glenn Thomas, Hoss Gifford, Manuel Clement, Andreas Heim, Robert Hodgin, Margaret Carlson, Erik Natzke, Andries Odendaal, James Tindall, Jon Williams, Ferry Halim, Jobe Makar, Jared Tarbell, Geoff Stearns, Paul Szypula, Lynda Weinman, the beta readers listed earlier, and whomever I've inevitably omitted.

Many thanks and much love to my wife, Wendy Schaffer, to my parents, and to family and friends. Hopefully this edition wasn't as draining as the first.

And lastly I'd like to thank you, the reader, for taking the time to read this book. I hope it helps to make my passion your own.

—Colin Moock
Toronto, Canada
December 2002