

1

Getting Started

Introducing ActionScript 3.0	3
Why You Should Learn ActionScript 3.0	3
What's New in ActionScript 3.0?	4
Differences Between ActionScript 1.0/2.0 and 3.0	4
ActionScript 3.0 Elements	6
Moving Beyond Script Assist	10
Caution: Player Required!	11
Beyond ActionScript 3.0	11

You may already be an accomplished Adobe Flash user, but creating fully interactive presentations requires using ActionScript 3.0, the internal programming language in Flash CS3 Professional. It's similar, but not identical, to Java. However, you do not have to know Java or JavaScript or be a programmer to include ActionScript in your movies. Though it is a little more difficult to learn than ActionScript 2.0, ActionScript 3.0 runs faster (in many cases), is more consistent, and overall is more powerful than ActionScript 2.0.

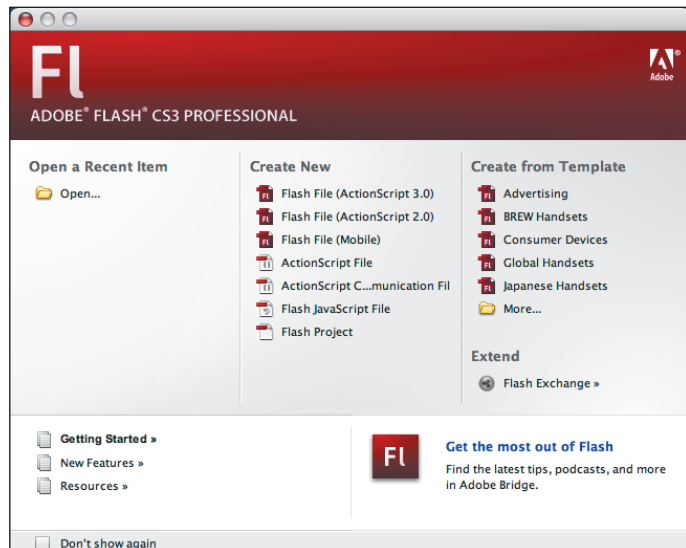
ActionScript is important because you can't accomplish many basic Flash activities—such as stopping a movie, restarting a movie, or controlling audio volume—without it. ActionScript also extends the power and flexibility of your project by letting you navigate the main **Timeline**, link to other URLs (**U**niform **R**esource **L**ocators) on the Internet, load other movies into a Flash CS3 movie, and do much more. By the time you are finished with this chapter, you will have a solid understanding of how to add ActionScript to your Flash movies. You will also learn most of the “must-know” ActionScript required for your own Flash projects.

Working with ActionScript code is one of the most technically challenging aspects of Flash CS3. This book will not teach you everything there is to know about ActionScript, but it will cover the basics and give you a solid foundation on which to build.

Introducing ActionScript 3.0

ActionScript 3.0 is an object-oriented programming language used to power the Flash Player. It's an ECMAScript language, similar to JavaScript, so if you want to learn other ECMAScript languages, ActionScript 3.0 is a great place to start. The latest version of ActionScript brings it into full compliance with ECMA standards.

ActionScript 3.0 can be embedded in a Flash project (.fla) file in Flash CS3, written as a stand-alone ActionScript (.as) file, or created in Flex Builder, a new tool built on Adobe's Flex framework that offers developers an environment for creating RIAs (Rich Internet Applications). This book will focus on using ActionScript 3.0 to enhance your Flash CS3 projects.



Why You Should Learn ActionScript 3.0

If you're a Flash designer, chances are you've gotten only so far using Script Assist. Learning ActionScript 3.0 will help you to leap that hurdle and create fully interactive applications, including dynamic Web applications and interactive video games. If you have learned ActionScript 1.0 or 2.0 and are intimidated by the language differences

in ActionScript 3.0, consider that ActionScript 3.0 has several key benefits over the previous version, including fast downloading speed, precise visual control, advanced interactivity, the capability to combine bitmap and vector graphics and include video or animation, and scalable and streaming content.

NOTE:



Download Speed

ActionScript is executed by the AVM (ActionScript Virtual Machine) in the Flash Player. ActionScript 3.0 introduces AVM2, which results in download speeds up to 30 times (30 times!) faster than legacy versions.

What's New in ActionScript 3.0?

ActionScript 3.0 offers a vast array of new features. The following chart outlines some of the new features:

ActionScript 3.0 New Features	
Feature	Description
Packages	ActionScript 3.0 classes are organized into packages , folders that hold similar ActionScript class files. Packages existed in ActionScript 2.0 but are used much more in ActionScript 3.0.
Document class	Flash CS3 has introduced something called a document class . In previous versions of Flash and ActionScript, the main Timeline was always a movie clip symbol. Now, you can create your own custom class for the main Timeline .
ActionScript tools	Flash CS3 has many new ActionScript tools in the Actions panel that help you learn how to write and organize code more effectively than ever. There are now many buttons to create comments and collapse blocks of code.
Scripting improvements	Flash CS3 now includes a new ActionScript debugger that offers improved flexibility and feedback and is consistent with Adobe's Flex 2 debugger. You can also convert animations directly to ActionScript and copy and paste ActionScript animation properties from one object to another.
Language consistency	If you are familiar with ActionScript 2.0, you may have noticed some inconsistencies in the language. ActionScript 3.0 is far more consistent in syntax, so the language is much more intuitive once you know the core concepts.

Differences Between ActionScript 1.0/2.0 and 3.0

ActionScript 3.0 includes many key differences, as you can see in the previous chart. Here's a practical example that will illustrate these differences more clearly.

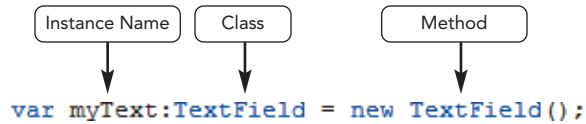
Let's say you want to add a text field to your Flash document. Using ActionScript 2.0, the code would look like this:

The diagram consists of four rounded rectangular boxes at the top, each with an arrow pointing down to a corresponding part of the code below. The boxes are labeled 'Class', 'Method', 'Instance Name', and 'Properties'. The code is: `myMovieClip.createTextField("thickness_text", 10, 0, 0, Stage.width, 22);`

```
myMovieClip.createTextField("thickness_text", 10, 0, 0, Stage.width, 22);
```

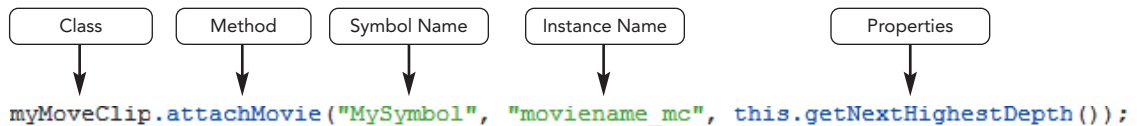
In the first line of code, you call the movie clip class. In legacy versions of ActionScript, all components had to be embedded in movie clips. Next, you use the `createTextField` method to add the text field. An instance name, in parentheses, is required for all components when you use ActionScript. The text field properties, the width and height and positioning on the **Stage**, follow the instance name.

In ActionScript 3.0, the same code would look like this:



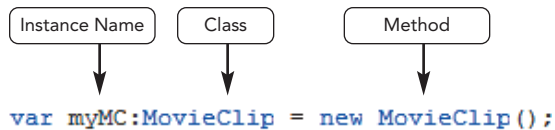
In ActionScript 3.0, you simply create a new instance of the text field class. You can supply the text field properties later.

Here's another example. The following line of ActionScript 2.0 code imports a movie clip symbol from your **Library** panel into the SWF file at run time:



Again, you must call the movie clip class, the **attachMovie** method, the symbol's name, the new instance name, and a depth value.

In ActionScript 3.0, you simply create a new instance of the movie clip class. You can add the properties, such as the instance name, symbol name, and **Stage** depth, later. The syntax is basically the same to create a text field or a movie clip.



Note this is just an isolated example. Not every function in ActionScript 3.0 requires less code than in previous versions. ActionScript 3.0 isn't necessarily a simpler language but, more important, is standardized. It doesn't matter whether you're creating a text field, a movie clip, a shape, or any other object. This standardization makes it easier to learn and easier to use.

ActionScript 3.0 Elements

In this chapter, you will learn about several core ActionScript 3.0 elements, including the following:

Variables

Instances

Properties

Functions and methods

Events, event handlers, and event listeners

Classes

Conditional statements

These are not *all* the elements of ActionScript 3.0, but knowing how to use these elements will allow you to do most of the common ActionScript tasks. The following sections can act a reference for you to understand each of the ActionScript 3.0 elements used in this book. If you get to a section in a later chapter that seems difficult or confusing, return to this chapter and review these terms.

Variables

Variables are containers that hold data. To understand variables, think of a game where the player has a score. The information (or **data**) about how many points the player has is contained in a variable. When the player gets more points, the number in the **score** variable increases. Thus, the **score** variable acts as a container (or variable) for a number (or data).

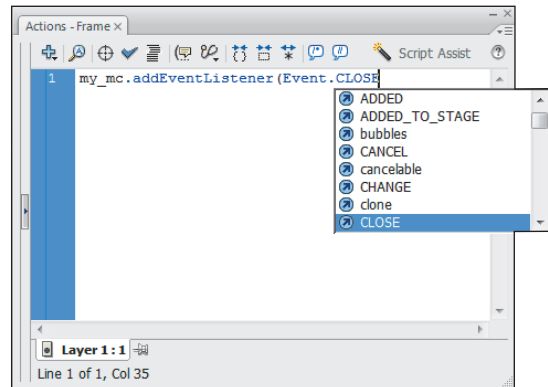
The data in a variable is not limited to numbers only. Variables can hold many types of data. Variables can hold text values, such as a user name, password, or text in a text field. They can also hold **true** or **false** values, such as whether or a user logged into a Web site has administrator status. The type of data a variable holds is called its **data type**.

In ActionScript 3.0, you create variables using the keyword **var**. The code to create a variable called **score** is **var score**.

Whenever you create a variable in ActionScript 3.0, you must give the variable a data type. To tell Flash CS3 the type of data a variable will hold,

type a colon and then the data type. Most data types begin with a capital letter. The code to tell Flash CS3 that the **score** variable will hold a **Number** data type is **var score:Number**.

The name of the data type that holds text is **String**, and the name of the data type that holds **true** or **false** values is **Boolean**.



Note: If you typed code such as that shown in the illustration here in the **Script** pane in the **Actions** panel, you may have noticed that after you typed the colon, a small menu appeared. This menu is called the **Code Hint** menu and is a useful tool in writing ActionScript 3.0.

You give a value to a variable using an equals sign. You can do this on the same line as you create your variable. The code used to assign a value of **0** to a variable called **score** is **var score:Number = 0;**. The code to create a variable called **text** with a value of "This is my text" is **var text:String = "This is my text";**.

Note: Notice the quotation marks around the text variable's value. Anytime you use the String data type, the value must be in quotation marks, or Flash CS3 will think you are referencing a variable. All other data types do not use quotation marks.

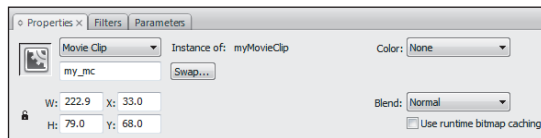
Also notice the semicolon after the Number value in **var score:Number = 0;**. A semicolon in ActionScript 3.0 is similar to a period in a sentence. Both denote the end of a statement.

Many variables do not contain the same value forever. Consider a player's score again—it may start at 0, but as the player gets more points, the value can change to 100; 500; or 1,948,762. To assign a new value to a variable that has already been created, do not use the **var** keyword. Instead, on a different line of code, type the variable name, an equals sign, and then the new value (followed by a semicolon, of course). The code to assign a value of **100** to a variable called **score** that was initially created on a previous line of code with a value of **0** is **score = 100;**

Instances and Instance Names

You are already familiar with instances. When you drag a symbol from the **Library** panel in Flash CS3, you are creating an **instance** of that symbol. One symbol can have many instances, but each instance is linked to only one symbol.

In Flash CS3, you must name instances if you want to communicate with them via ActionScript. For example, if you want your Flash movie to stop playing when you click a button, you must give that button an instance name. You can give instance names to movie clips, buttons, and text fields but not to graphic symbols. To give an instance name to an instance, select the instance, and type an instance name in the **Instance Name** field in the **Property inspector**:



In ActionScript, you refer to individual instances by their instance names, not by their symbol names in the **Library** panel. The name you give your instances is important and is similar to how you must name your symbols. Always start with a lowercase letter, and do not use spaces or special characters other than underscores. It is a best practice to pick a naming convention and use it consistently. Instance names should also be descriptive but brief (so you don't have to type more than necessary). Some examples of good instance names are **contact_btn**, **nav_mc**, and **body_txt**.

Note: If you end your instance names with **_mc** for movie clips, **_btn** for buttons, and **_txt** for text fields, Flash CS3 will give you access to code hinting in the **Actions** panel. This makes writing code much easier and faster.

Properties

Believe it or not, you are already familiar with properties by this point. Properties are simply variables that are attached to an instance of a symbol. If that sounds a little cryptic, think of properties as attributes you can modify in the **Property inspector**. Specifically, the **x** and **y** positioning and the width and height of an instance are some of its properties. Working with movie clips or buttons, some other properties include alpha, blend mode, and even filters. You can modify properties in ActionScript using **dot syntax**, which means you type the instance name, a dot, the property you want to modify, and then the value you want to give that property using an equals sign (just like setting the value of a variable). The code to modify the X position of a movie clip with an instance name of **my_mc** and set it to **100** is **my_mc.x = 100;**

Functions and Methods

When using ActionScript 3.0, you will often write large blocks of code that you reuse many times. Copying the same code several times can be frustrating and tedious. Fortunately, you can write a block of code one time and recycle it using functions and methods. Functions and methods are essentially the same—both are reusable blocks of code.

Many functions already exist in Flash CS3. If you want to stop a movie from looping continuously, you can use the prebuilt **stop()** function. To use a function, type the name of the function, and then type a pair of parentheses. The code to run the **stop()** function is **stop();**. Another common prebuilt function in Flash CS3 is **gotoAndPlay()**. The **gotoAndPlay()** function plays the Flash movie starting at a particular frame. When you run the **gotoAndPlay()** function, you tell Flash CS3 the frame you want to play in the parentheses. For example, the code to play a movie from the fifth frame is **gotoAndPlay(5);**

Note: The parentheses are what make the function run, and they differentiate a function from a variable. If you just typed **stop** with no parentheses, Flash CS3 would think you were referring to a variable called **stop**.

Some functions do not already exist in Flash CS3, so you will have to create them. Writing custom functions requires a few steps. First, you must tell Flash CS3 you are creating a function by using the **function** keyword. Next, you tell Flash CS3 the name of your function. If you wanted to create a function called **myFirstFunction**, you would type **function myFirstFunction**. The next step is to type a pair of parentheses. You will learn more about what the parentheses are for when you learn about events. After the parentheses, you need to tell Flash CS3 the return data type using a colon (similar to how you set a data type for a variable). The return data type for all functions used in this book will be **void**. Last, you put all the code block that you will reuse inside curly braces. For example, the code to create a complete function called **myFirstFunction** is as follows:

```
function myFirstFunction():void
{
    (Block of code goes here)
}
```

Running a custom function is similar to running a prebuilt function. Simply type the name of the function and then the pair of parentheses. The code to run a function called **myFirstFunction** is **myFirstFunction()**;

Events, Event Handlers, and Event Listeners

Events are things that happen while a Flash movie is playing. Many types of events exist, such as when a visitor to your Web site clicks a button, presses a key on the keyboard, or starts downloading a file. You can utilize events by running functions when events happen. The special functions that run when events happen are called **event handlers**.

To write an event handler, simply create a basic function. The only difference with an event handler

function is that it receives information about the event that made the function run. The code to create an event handler function called **playMovie** that reacts to a button click is as follows:

```
function playMovie(event:MouseEvent):void
{
}
```

Note: The **event:MouseEvent** code in the parentheses is significant. This is how you capture information about what caused the function to run. The **event** part represents the event that happened, and the colon specifies the data type of this event, which is **MouseEvent**. Before this function will run when you click a button, you need to attach the event handler function to the event (the button click).

To attach an event to an event handler, you need to use something called an **event listener**. Event listeners wait for events to happen, and when the events happen, the appropriate event handler function runs. To understand event listeners, think of a radio station broadcasting music as an event. It broadcasts whether or not you listen to it. Tuning your radio to a station is like listening for an event. When you hear the music, you choose how to react to it (that is, sing along, change the station, or turn the volume up). In the same way that your selection of a radio station connects you to the signal being broadcast, event listeners connect event handlers to events.

To have a button or any object in Flash CS3 listen for an event, use the **addEventListener** method (**method** is synonymous with **function**) of any object. You do this by typing the instance name, typing a dot, and then typing **addEventListener**. Then in parentheses, tell Flash CS3 the event the instance is listening for, type a comma, and then type the name of the function that will run when the event happens. For example, if you had a button with an instance name of **play_btn** and you wanted to run a function called **playMovie** when you clicked it, you would type the following:

```
play_btn.addEventListener(MouseEvent.CLICK,
playMovie);
```

Note: In Flash CS3, events start with the data type of the event (in this case **MouseEvent**), then have a dot, and then have a specific event name in all caps. Also note that anytime you send multiple values to a function, you separate the values with commas.

Classes

Classes might be somewhat familiar to you already. A **class** is a blueprint, or concept, of something. Think of movie clip symbols and button symbols. What are some differences between the two? Movie clips have nearly an unlimited number of frames, and they play when your Flash movie is running unless you use ActionScript to tell them otherwise. Buttons have a **Timeline** with only four frames and do not play unless you roll over or click them. The **MovieClip** class is the blueprint for all movie clips. Though each movie clip symbol may look different, all movie clips have certain similarities. Classes all begin with a capital letter, and all classes are also data types. **Number**, **String**, and **Boolean** are all classes.

When you drag instances of movie clip or button symbols out of your **Library** panel and put them on the **Stage**, you are creating instances of the **MovieClip** or **Button** class. Many classes are not visual, so you must create instances of those classes with ActionScript using the **new** keyword. After the **new** keyword, type the class name and a pair of parentheses. One nonvisual class is called **Loader** and is used to load external content. To create a new instance of the **Loader** class called **myLoader**, you would type the following:

```
var myLoader:Loader = new Loader();
```

Note: This code creates a new instance of the **Loader** class and is similar to creating instances of many other nonvisual classes. After the **new** keyword, notice the name of the class and the pair of parentheses. This looks similar to the syntax used to run a function—because it *is* a function. It's a special function called a **constructor function** that creates a new instance of that particular class.

Conditional Statements

Conditional statements allow you to run a block of code based on a condition being true or false. Picture yourself getting dressed in the morning (or whatever time of the day you prefer to get dressed). Assuming you are planning to wear both shoes and socks that day, you put your shoes on only if you are already wearing socks. If you don't have socks on, then you put socks on and then shoes.

In ActionScript 3.0, conditions are computed in a similar way. Consider a variable called **socksOn** with a **true** or **false** data type (Boolean). To write a conditional statement, you first use the keyword **if**, then you place the expression that is evaluated as **true** or **false** in parentheses, and finally you place the code that executes in curly braces. In code, a conditional statement that would put shoes on if socks were already on would look like this:

```
if(socksOn)
{
    (Put shoes on)
}
```

Note: If the **socksOn** variable had a value of **true**, the code that says **(Put shoes on)** would run. Everything in the parentheses after the keyword **if** is evaluated as **true** or **false**.

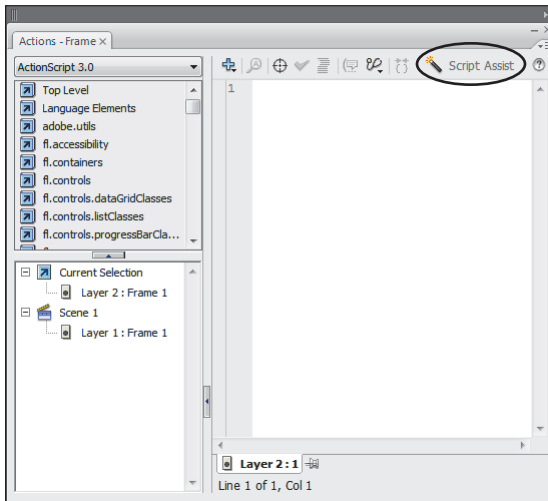
You can also run a block of code if a condition is not true. To do that, use the keyword **else**, and place the code you want to run in curly braces. Using the previous code, if you wanted to put shoes on if **socksOn** were true and put socks on if **socksOn** were false, you would type the following:

```
if(socksOn)
{
    (Put shoes on)
}
else
{
    (Put socks on)
}
```

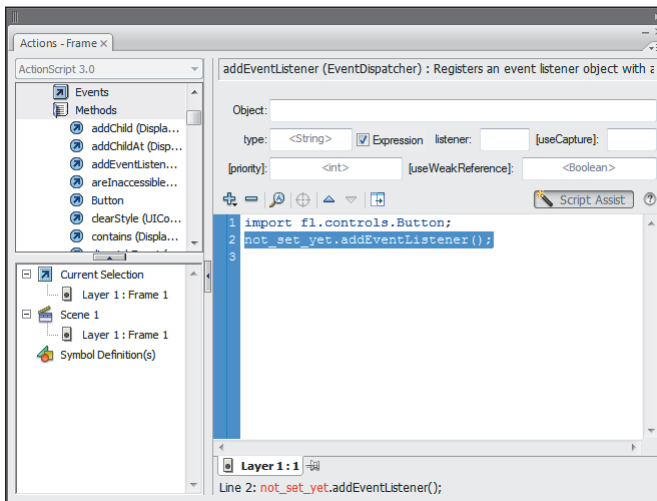
Note: You don't need any parentheses after the **else** keyword.

Moving Beyond Script Assist

If you've done a lot of Flash design, chances are you have used Script Assist, which is a feature in Flash that helps you write ActionScript. It can be helpful when you know the functions or methods you need to call but can't remember the correct syntax. To turn on Script Assist, open the **Actions** panel by choosing **Window > Actions** or by hitting **F12** (Windows) or **Opt+F12** (Mac) on your keyboard and clicking the **Script Assist** button in the upper-right corner of the screen:



Double-click any method, and you'll be prompted to type the required properties:



Script Assist is a great tool for writing simple ActionScript. However, if you don't have a solid understanding of ActionScript to begin with, Script Assist is not going to work very well for you. This book will lay the foundation that you need to write solid, concise ActionScript, and you may very well find that as your skills improve, you may move beyond Script Assist. When you create the code yourself, you can add advanced interactivity and add more power to your applications.

Caution: Player Required!

Flash CS3 content is not visible in a browser unless either the Flash Player or the Shockwave Player has been installed in that browser. In the past, this was seen as a serious limitation of the format, although over the past few years the number of Internet users who have the player has increased exponentially because current browsers now come with the Flash Player preinstalled. Shockwave was designed to view content created with Adobe Director and has not reached quite the same level of popularity as Flash.

Adobe has hired an independent consulting firm to maintain an estimate of the number of Flash Players in use. At the time of this writing, the Flash Player is installed on 98 percent of Internet-enabled desktops globally, and more are 200 million Flash-enabled mobile devices are in existence. Flash Player 9 comes preinstalled on all new browsers shipped by AOL, CompuServe, Microsoft, and Netscape. Additionally, all versions of Microsoft Windows 98 and newer and Apple OS 8 and newer include the plug-in.

The following chart describes the two players:

Adobe Players	
Description	Player
Flash Player	The Flash Player is used for viewing Flash content on the Web. You can download the latest version of the Flash Player at www.adobe.com/downloads/ . This player installs in the player folder for your browser of choice.
Shockwave Player	The Shockwave Player is used for viewing Director content on the Web. You can download the latest version of the Shockwave Player at www.adobe.com/downloads/ .

Beyond ActionScript 3.0

Flash CS3 is an incredibly powerful tool by itself. However, it can't perform a few functions. The following sections present some of the Web technologies you should know about if you want to extend Flash CS3 beyond its basic capabilities. Using these technologies is beyond the scope of this book. However, it is good to understand what these technologies are, especially if you plan to further your ActionScript skills when you are finished with this book.

What's AIR?

Adobe Air (Adobe Integrated Runtime), formerly code-named Apollo, is a cross-operating system runtime, or a framework that works along with the

operating system to support applications written specifically for the framework. C++ is an example of an existing popular runtime. For years, Web developers have been developing RIAs for the Web. With AIR these developers can use the programming knowledge they already have (such as HTML, Flash, Ajax, and so on) to build and publish desktop applications. Unlike Web applications, AIR applications will not have to run within a browser environment and can support traditional desktop application features, such as drag-and-drop interactivity, desktop shortcuts, Clipboard access, and integration with other desktop

applications. It's an exciting prospect and one that Flash users are eagerly anticipating.

For further information about AIR, please check out the following URLs:

<http://labs.adobe.com/technologies/air/>

www.codeapollo.com/

What's CGI?

A CGI (**C**ommon **G**ateway **I**nterface) script is a program that defines a standard way of exchanging information between a browser and a server. You can write CGI scripts in any number of languages (Perl, C, ASP, and others). If you plan to create a complex Web application that requires using something like CGI, it is recommended that you work with a Web engineer who has experience creating these kinds of scripts. Flash CS3 can communicate with CGI scripts, although that topic is beyond the scope of this book.

For further information about using CGI, please check out the following URLs:

www.cgidir.com/

www.cgi101.com/

www.ichthus.net/CGI-City/

What's XML?

XML (**eX**tensible **M**arkup **L**anguage) is a standard that handles the description and exchange of data. With XML, developers can create markup languages that define the structure and meaning of information. Therefore, an XML document is much like a database presented in a text file. You can transform XML content into a variety of formats, including HTML, WML (**W**ireless **M**arkup **L**anguage), and VoiceXML.

XML differs from HTML in that it is not predefined—you create the tags and attributes. You can also use XML to create your own data structure and modify it for the data you want it to carry. In Flash CS3, you can use the XML object to create, manipulate, and pass that data. Using ActionScript, a Flash CS3

movie can load and process XML data. As a result, an XML-savvy Flash CS3 developer can develop a movie that dynamically retrieves data from the external XML document instead of creating static text fields within a project file.

Just as HTML provides an open, platform-independent format for distributing Web documents, XML has become the open, platform-independent format for exchanging any type of electronic information. Like CGI, XML is also a topic beyond the scope of this book.

For further information about XML, take a look at the following URLs:

www.ait-usa.com/xmlintro/xmlproject/article.htm

www.xml.com/

www.xmlfiles.com/

JavaScript and Flash CS3

ActionScript is based on JavaScript, another scripting language. Although ActionScript and JavaScript share a similar syntax and structure, they are two different languages. One way to tell them apart is that ActionScript uses scripts processed entirely within the Flash Player, independently of the browser used to view the file. JavaScript, on the other hand, uses external interpreters that vary according to the browser used.

You can use ActionScript and JavaScript together because Flash CS3 lets you call JavaScript commands to perform tasks or to send and receive data. A basic knowledge of JavaScript can make learning ActionScript easier, because the basic syntax of the scripts and how objects are handled are the same in both languages. However, JavaScript is not a requirement for learning ActionScript.

For further information and tutorials about JavaScript and how to use it in conjunction with Flash CS3, check out the following URLs:

www.javascript.com/

<http://javascript.internet.com/>

www.flashkit.com/links/Javascripts/

That's a wrap for this chapter. You've familiarized yourself with the basic concepts of ActionScript 3.0, the differences between 3.0 and legacy versions, and how to extend Flash content even further using various technologies. Now it's time to dive into the details of ActionScript 3.0 and start writing some actual code. On to the next chapter!
