Part I

Getting Started with ActionScript 3.0

IN THIS PART

Chapter 1

Introducing ActionScript 3.0

ActionScript 3.0 is vastly new and different and at the same time remarkably familiar for those who have worked with ActionScript 2.0. ActionScript 3.0 is new because it's built from the ground up, based on a new edition of the ECMA specification, and runs in a new virtual machine (AVM2). Yet despite all the newness of ActionScript 3.0, it is still similar enough to ActionScript 2.0 that those who have previously worked with Flash will find their footing without much difficulty.

In this chapter we look at what ActionScript is, where you can use it, and the specifics of what's new to ActionScript 3.0.

IN THIS CHAPTER

What Is ActionScript?

Presumably, if you're reading this book, you already have a general sense of what ActionScript is, but further clarification will help you to better understand how to build content using ActionScript. In a nutshell, ActionScript is a programming language that is used to create content for Flash Player. You can use tools such as Flash CS3 Professional or Flex Builder to create content using other tools and technologies such as drawing tools, library symbols, timelines, and MXML. However, ActionScript can be used either as a complement to these things or in place of these things in order to create Flash content. ActionScript is often necessary when you want to create Flash applications that are highly dynamic, responsive, reusable, and/or customizable. Here's just a short list of the many things you can accomplish using ActionScript:

- Loading images
- Playing audio and video

- Drawing programmatically
- Loading data such as XML files
- Responding to user events such as mouse clicks

Where Can You Use ActionScript?

At the time of this writing there are two primary ways to work with ActionScript: building Flash applications using Flash authoring (Flash CS3) or building Flex or ActionScript applications using Flex Builder 2. There is no black-and-white rule for when or why to use one authoring toolset over the other. Rather, there is a large gray area of overlap between the two. A helpful way to decide which toolset is most appropriate for you is simply to consider what workflow you're more accustomed to using. If you are most comfortable using drawing tools and timelines, then Flash CS3 is likely the best option. On the other hand, Flex Builder 2 is good for those who feel most comfortable writing code in a robust IDE (integrated development environment) and don't feel comfortable with the animation metaphors used by Flash CS3. With all that said, there is no reason that you couldn't use both authoring toolsets for building applications. The good news in this case is that ActionScript 3.0 is identical regardless of which toolset you use. We have made an effort to write this book in such a way that you can use the examples with either Flash CS3 or Flex Builder.

If you are using Flex Builder 2 (note that we don't discuss the Flex framework in this book) to create applications using ActionScript then you are forced to use strict object-oriented design, meaning that the main entry point for your application must be a class (more on classes in Chapter 3). This is not so for Flash CS3. If you are using Flash CS3, then you have the option to place your ActionScript code on keyframes on a timeline. While this is not inherently wrong, it is troublesome if the goal is to write good code that is maintainable and adheres to best practices. For this reason, in this book we advocate for the use of a document class if you are using Flash CS3, and all the examples in this book use document classes. In Flash CS3, the document class is the analog to the main class in a Flex Builder 2 project. By using a document class you will be learning the most scalable way to write ActionScript 3.0 code in Flash CS3, and what you learn will easily translate if you use Flex Builder 2 at any time.

What's New in ActionScript 3.0?

As mentioned earlier in this chapter, ActionScript 3.0 has a whole lot of new features. You'll learn about all the details in the chapters that follow, but here is an overview of the key new features.

Display List

In ActionScript 2.0, there were three basic types of objects that could be displayed: movie clips, buttons, and text fields. These types didn't inherit from a common source, meaning polymorphism didn't work for these display types. Furthermore, instances of these display types always had a

fixed, parent-child relationship with other instances. For example, to create a movie clip you had to create that movie clip as a child of an existing movie clip. It was not possible to move a movie clip from one parent to another.

In ActionScript 3.0 there are many new display types. In addition to the familiar types such as movie clips, buttons, and text fields, you'll now find new types such as shapes, sprites, loaders, bitmaps, and more. All display types in ActionScript 3.0 inherit from `flash.display.DisplayObject`, meaning you can use polymorphism. Furthermore, and perhaps most important, display type objects in ActionScript 3.0 can be created independent of any other display type object, and these objects can be placed as children of other display objects and even moved from one parent container. In other words, you can create a text field in ActionScript 3.0 simply by calling the constructor as part of a new statement, and that text field will exist independent of any parent container object.

```
var text:TextField = new TextField();
```

You can then add the text field to a parent container at any time. The following example illustrates this with a display object called `container`, which could be a sprite or any other display object container type:

```
container.addChild(text);
```

NOTE **In the preceding example, `container` is used as a generic variable name that would presumably refer to an object created elsewhere in the code.**

The hierarchy of parent containers and their children is known as the *display list* in ActionScript 3.0.

Runtime Errors

ActionScript 3.0 provides many new runtime errors. This is an important new feature because it allows you to diagnose problems much more quickly. In ActionScript 2.0, when an error occurred at runtime it would frequently occur silently, and it would be difficult for you as the developer to determine what the problem was. With improved runtime errors and error reporting in the debug player it is now much easier to debug ActionScript 3.0 applications than it was with ActionScript 2.0.

Runtime Data Types

Strict typing in ActionScript 2.0 was only used by the compiler, not at runtime. At runtime, all ActionScript 2.0 types are dynamic. However, in ActionScript 3.0, strict typing is preserved at runtime as well. The advantage is that now runtime data mismatches are reported as errors, and application performance and memory management is improved as a result of preserved typing at runtime.

Method Closures

In ActionScript 3.0 all methods have proper method closures, which means that a reference to a method always includes the object from which the method was originally referenced. This is important for event handling, and it stands in stark contrast to method closures in ActionScript 2.0. In

ActionScript 2.0, when you reference a method, the object from which the method is referenced does not persist. This causes problems most notably when adding event listeners. In ActionScript 2.0, a delegate is often used as a solution. However, in ActionScript 3.0 delegates are not necessary.

Intrinsic Event Model

In ActionScript 3.0, the event model is built right in to the core language. The `flash.events.EventDispatcher` class is the base class for many native ActionScript classes, including all the display object types. This means that there is one standard way to dispatch and handle events in ActionScript 3.0.

Regular Expressions

Regular expressions are a powerful way to find substrings that match patterns. Although regular expressions have long been built into sister languages such as JavaScript, regular expressions were never a part of ActionScript until now. ActionScript 3.0 includes an intrinsic `RegExp` class, which allows you to run regular expressions natively in Flash Player.

E4X

E4X is short for ECMAScript for XML, and it is a new way to work with XML data in ActionScript. Although you can still work with XML as you did in ActionScript 2.0 by traversing the DOM, E4X allows you to work with XML in a much more natural and intuitive manner.

Summary

- ActionScript 3.0 is a new language with enough similarities to ActionScript 2.0 to make the learning curve low.
- You can use ActionScript 3.0 just as you used ActionScript 2.0 (in classes or on the timeline), though the preferred usage of ActionScript 3.0 is in classes using object-oriented principles.
- ActionScript 3.0 introduces a lot of new features, including a new way to manage display types, runtime error handling, runtime data types, method closures, an intrinsic event model, regular expressions, and a new way of working with XML.