

Flasheezy.com Presents



Stepping Into ActionScript 3

By Adam Petrie

The complete guide for easy migration from AS2 to AS3

© **Flasheezy.com 2009**

All rights reserved. No part of this publication may be reproduced or redistributed in any form without the prior written permission of the publishers.

Table of Contents

Introduction	4
What You Need To Know	5
What You Will Learn	6
A Brief History of ActionScript	7
Variables	8
Functions	13
Interfaces & Classes	18
Interfaces	25
Events	28
Display Lists	30
Sample Application	33
Additional Resources	37
About The Author	38

Introduction

Learning a new development language is not always easy. Like traditional languages, each development language comes with its own grammar, punctuation and nuances that can prove to be very frustrating to master. Similarly, in order to gain fluency in any language, you need to use it. Learning about the language is not enough and practice truly does make perfect.

ActionScript 3 (AS3) is no exception to this rule. With a more advanced and complex structure than its predecessors, AS3 has a steep learning curve that can be very intimidating to the beginning programmer. Once mastered, however, AS3 is also faster, more powerful and easier to use than ActionScript 2. This is why, in the following pages, it is my hope that you will find everything you need to learn and practice the fundamentals of ActionScript 3 development.

We'll start with the most basic building blocks in any language, variables. We'll discuss how they work and what you need to know to make them work for you. From there will kick it up a notch and learn about functions. Functions will take our knowledge of variables and help us create some useful code snippets. The next step will be classes and interfaces. We'll learn how to use them, why they are amazing and we'll look at some simple examples to illustrate the general concept.

With variables, functions and classes in our toolkit, I'll explore two concepts that are in some ways very unique to AS3 - Events and the Display List. Events will help us add interactivity to our projects and the display list will teach us how to manipulate objects on our stage. Finally, we'll wrap the whole process together with a sample application that will make use of everything we've learned. However, before we get into that, let's make sure you've got the necessary background information to be able to follow along.

What You Need to Know

Stepping into ActionScript 3 is a beginner's guide to development with AS3 for the Flash and Flex development environments. Previous ActionScript knowledge is not required and due to the drastic changes in AS3 from ActionScript 1 and 2, in some cases, previous knowledge may not be beneficial. Together we will cover standard AS3 usage from the ground up so you can better understand how the language works and how you can use it in your own projects.

While ActionScript knowledge is not required, a basic understanding of coding principles will not be covered in this book. In effort to focus on the specifics of the language, I will not be covering standard development constructs, but rather how to use standard constructs inside of AS3. In addition, this book does not cover basic Object Oriented Programming (OOP) principles.

For the same reasons as above, the book will focus on how to apply OOP to ActionScript 3 instead of explaining how OOP principles work. Lastly, it is important to note that while this book teaches the development language used inside of Adobe Flash and Adobe Flex, I will not discuss using either of these environments in detail. My aim is to teach AS3 outside of a specific context and, as such, there is little to no assumed knowledge with these specific tools.

What You Will Learn

By reading through the contents of this book, it is my goal to leave you with the following information:

- **Correct usage of AS3:**

The main goal of this book to offer you the information you need to code in AS3 quickly and easily. It is my hope to arm you with information that will allow you to step into the API and learn new concepts using your familiarity with the basics as a starting point.

- **Best Practices:**

With any particular development problem there are 100's of possible solutions. Some of those solutions however are much better than others for various reasons. Throughout the book you will find best practices highlighted as a means to help you create optimal solutions in your work.

- **Easy migration from AS2 to AS3:**

While AS3 is vastly different from AS2, both languages still strive to accomplish the same tasks. The biggest difference is that AS3 simply does a much better job of allowing you to complete tasks than previous versions of ActionScript did. With the help of this book you should be able to migrate your knowledge of AS2 to AS3 with ease and trust me, once you've seen what AS3 has to offer, you'll never go back.

A Brief History of ActionScript

Before we begin it's important to take a quick look at where we are and where we've come from. Prior to ActionScript 3 came its predecessors ActionScript 2 and ActionScript 1. Both previous versions of ActionScript ran on the AVM1 (ActionScript Virtual Machine 1) and despite their differences during development, they essentially compiled down to the same language.

For the release of ActionScript 3, Adobe introduced an entirely new AVM (AVM2) which was designed and built specifically for executing the ActionScript 3 language. The result of this change means that AS1 or AS2 code cannot directly communicate with AS3 code and that the jump from ActionScript 2 to ActionScript 3 is a big one.

It's not all doom and gloom though. In fact once you've learned ActionScript 3 you'll find it's quite the opposite. While common tasks often require more code in ActionScript 3, thanks to the new virtual machine, the same tasks are executed much faster and they are more often than not easier to debug. How much easier? Let's move onto basic variables and you'll find out.

8

Variables

As most of us know, variables are essential to code as they allow developers to store data and values. In ActionScript 3 defining a variable is performed with the one of the following lines of code:

```
1 var myVar;  
2 var myVar2:varType;  
3 var myVar3:varType = varValue;
```

Each of the lines above defines a different variable and each one of them defines the variable in a slightly different way. Each example above has two things in common:

- The var keyword
- The semicolon at the end of each line

The keyword var is used to denote a variable definition and the semicolon is used to indicate the end of a single command to the compiler.

In the first example we are simply indicating that the term myVar is a variable. It is the simplest way to define a variable with AS3. In the second example we've not only defined a variable but we've also given our variable the type varType. For now it is important to note only that variable types exist. We'll discuss them in greater detail later on.

The third example is the same as the second one except that after we define our variable we also provide it with an initial value. The third scenario while quite common is not required to initialize or store new data into a variable. Continuing from the second example above, when we want to store new data in our defined variable, we use the following:

```
1 var myVar2 = value;
```

You'll notice above that we use the equals (=) operator just as before, but this time we omit the `var` keyword. Once our variables are defined, we no longer need to precede variable names with the keyword. In fact the compiler will display an error if we define the same variable twice. It is also important to note that regardless of how you define your variables based on the three original samples, you can always use the equal operator to change the data stored within them. The only caveat to this is that you cannot always store whatever you choose in any given variable. To explain this further, let's talk about variable types.

Variable types are used to tell the compiler what to expect from the data that is stored inside a variable. Variable types are nothing new with AS3; but what has changed is that in ActionScript 2 variable types were only used for compile-time checking whereas AS3 also enforces data types during movie playback. Essentially this change means that AS3 is much more strict when it comes to storing data in a variable and that you'll need to be mindful of your type declarations.

ActionScript 3 supports the following variable types:

- String
- Boolean
- Number
 - int
 - uint
- Object types
- Undefined type

String variables are used to store groups of alphanumeric characters that are enclosed with quotations.

```
1 var myCat:String = "Mittens";
```

Boolean variables are used specifically for holding values that can only be true or false as follows:

```
1 var catsLikeMilk:Boolean = true;  
2  
3 var catsLikeDogs:Boolean = false;
```

Numbers are used to store numerical data such as integers and decimals.

```
1 var catsAge:Number = 3;
```

While String, Boolean and Number all existed in ActionScript 2, int and uint are new to ActionScript 3. Essentially int and uint are special types of Numbers and as such they are not capitalized like all the other variables types thus far. The keyword int stands for integer and int variables can only hold positive or negative whole numbers. The keyword uint stands for unsigned integer and may only be used to hold whole numbers that are not negative. Here are some more examples:

```
1 var catsAge:Number = 1.5;  
2  
3 var catsLives:uint = 9;  
4  
5 var catsLives:int = 9;
```

Note that while we could have stored `catsLives` inside a number we could not have stored `catsAge` in an int or uint because it contains a decimal. Considering that, you may be wondering why int and uint exist when Number covers all of the bases. The benefit of int and uint is that they are able to perform simple integer based math and are more efficient than Number.

The next type of variable to discuss is Object types. Objects are used to store groupings of data. For example, we may want an Object of type Array:

```
1 var myArray: ;
```

Not only does AS3 include a variety of built in Object types but whenever you define a class, you create a type definition as well. Let's assume we have defined a class call Cat.

```
1 var myCat:Cat;
```

For now, that will suffice, but if you really need to know more about Object types, skip ahead to the Classes & Interfaces chapter.

The last variable type on our list actually has no type at all. In the event that you do not want to provide a type for your variable, you can resort to the undefined type, which is denoted by an asterisk (*) in place of a data type. The undefined type allows you more freedom when storing data, but it is important to use this data type sparingly. Data types are meant to make your code more efficient and easier to read; while using the undefined type may sometimes be necessary, overuse of it could result in sloppy, confusing code.

In addition to variables, ActionScript also allows developers to define constants for values that we do not need to change.

```
1 const CAT_SPECIES:String = "Feline";
```

The difference here is that instead of using the var keyword we use the const keyword as well as following the general convention that constant names are all capital letters. Constants can be very useful when it comes to referencing static values and it is considered a best practice in favor of simply hardcoding the value when it is needed. The benefit of doing so is threefold:

1. If we need to change the value of the constant, we are able to update our code in one place (the constant definition) instead of having to change the value from "Feline" to "Canine" everywhere it has been used.
2. If we mistakenly type the wrong name of the constant (say KAT for example) the compiler will throw an error and we can easily fix the problem.

```
1 //Compiler error
2 var animalSpecies:String = KAT;
3 //No compiler error
4 var animalSpecies:String = "Feeline";
```

3. Using constants to denote values allows us to make these values more meaningful to anyone else who might read our code

```
1 //This date is meaningless to most people
2 var date:String = "May 26th, 1983"
3 //If it was defined as a constant named MY_BIRTHDAY, it might
4 //make more sense to others.
5 var date:String = "MY_BIRTHDAY"
```

Admittedly, the examples above are oversimplified, but believe me, using constants instead of 'magic numbers' as they are referred to can be a big help when someone else is reading your code, or you are debugging your own code.

Now that we are able to define variables and constants, we have all the tools we need to build a simple application; and we have discussed the building blocks that you will use throughout your code. The next topic we'll discuss is functions. We'll use them to automate certain tasks or groups of tasks and make our lives easier.