Action Script

Vasil Bozhurski Xia

Технологично училище "Електронни системи" към ТУ-София

www.elsys-bg.org

I. Introduction

ActionScript is an object-oriented language originally developed by Macromedia Inc. (now owned by **Adobe** Systems). It is a dialect of ECMAScript (meaning it is a superset of the syntax and semantics of the language more widely known as JavaScript), and is used primarily for the development of websites and software targeting the **Adobe Flash Player**platform, used on Web pages in the form of embedded SWF files. The language itself is open-source in that its specification is offered free of charge and both an open source compiler (as part of **Adobe** Flex) and open source virtual machine (Mozilla Tamarin) are available.

ActionScript was initially designed for controlling simple 2D vector animations made in **Adobe** Flash (formerly Macromedia Flash). Initially focused on animation, early versions of Flash content offered few interactivity features and thus had very limited scripting capability. Later versions added functionality allowing for the creation of Web-based games and rich Internet applications with streaming media (such as video and audio). Today, **ActionScript** is suitable for use in some database applications, and in basic robotics, as with the Make Controller Kit.

Flash MX 2004 introduced **ActionScript** 2.0, a scripting programming language more suited to the development of Flash applications.

It is often possible to save time by scripting something rather than animating it, which usually also enables a higher level of flexibility when editing.

Since the arrival of the **Flash Player** 9 alpha (in 2006) a newer version of **ActionScript** has been released, **ActionScript** 3.0. **ActionScript** 3.0 is an object-oriented programming language allowing far more control and code reusability when building complex Flash applications. This version of the language is intended to be compiled and run on a version of the **ActionScript** Virtual Machine that has been itself completely re-written from the ground up (dubbed AVM2). Because of this, code written in **ActionScript** 3.0 is generally targeted for **Flash Player** 9 and higher and will not work in previous versions. At the same time, **ActionScript** 3.0 executes up to 10 times faster than legacy **ActionScript** code. Flash libraries can be used with the XML capabilities of the browser to render rich content in the browser. This technology is known as Asynchronous Flash and XML, much like AJAX. **Adobe** offers its Flex product line to meet the demand for Rich Internet Applications built on the Flash runtime, with behaviors and programming done in **ActionScript**. **ActionScript** 3.0 forms the foundation of the Flex 2 API.

II. History

Action Script started as an object-oriented language for Macromedia's Flash authoring tool, now developed by **Adobe** Systems as **Adobe** Flash. The first three versions of the Flash authoring tool provided limited interactivity features. Early Flash developers could attach a simple command, called an "action", to a button or a frame. The set of actions was basic navigation controls, with commands such as "play", "stop", "getURL", and "gotoAndPlay".

With the release of Flash 4 in 1999, this simple set of actions became a small scripting language. New capabilities introduced for Flash 4 included variables, expressions, operators, if statements, and loops. Although referred to internally as "**ActionScript**", the Flash 4 user manual and marketing documents continued to use the term "actions" to describe this set of commands.

Table. 1. Timeline By Flash Version

Flash Player 2	The first version with scripting support. Actions included gotoAndPlay, gotoAndStop, nextFrame and nextScene for timeline control.
Flash Player 3	Expanded basic scripting support with the ability to load external SWFs (loadMovie).
Flash Player 4	First player with a full scripting implementation (called Actions). The scripting was a flash based syntax and contained support for loops, conditionals, variables and other basic language constructs.
Flash Player 5	**Flash Player** 5: Included the first version of **ActionScript**. Used prototype-based programming based on ECMAScript, and allowed full procedural programming and object-oriented programming.Design Based development

Flash Player 6	**Flash Player** 6: Added an event handling model, accessibility controls and support for switch. The first version with support for theAMF and RTMP protocols which allowed for on demand audio/video streaming.
Flash Player 7	**Flash Player** 7: Additions include CSS styling for text and support for **ActionScript** 2.0, a programming language based on theECMAScript 4 Netscape Proposal with class-based inheritance. However, **ActionScript** 2.0 can cross compile to **ActionScript** 1.0byte-code, so that it can run in **Flash Player** 6.
Flash Player 8	**Flash Player** 8: Further extended **ActionScript** 1/**ActionScript** 2 by adding new class libraries with APIs for controlling bitmap data at run-time, file uploads and live filters for blur and dropshadow.
Flash Player 9	**Flash Player** 9 (initially called 8.5): Added **ActionScript** 3.0 with the advent of a new virtual machine, called AVM2 (**ActionScript** Virtual Machine 2), which coexists with the previous AVM1 needed to support legacy content. Performance increases were a major objective for this release of the player including a new JIT compiler. Support for binary sockets, E4X XML parsing, full-screen mode and Regular Expressions were added. This is the first release of the player to be titled **Adobe Flash Player**.
Flash Player 10	**Flash Player** 10 (initially called Astro): Added basic 3D manipulation, such as rotating on the X, Y, and Z axis, a 3D drawing API, and texture mapping. Ability to create custom filters using **Adobe** Pixel Bender. Several visual processing tasks are now offloaded to the GPU which gives a noticeable decrease to rendering time for each frame, resulting in higher frame rates, especially with H.264video. There is a new sound API which allows for custom creation of audio in flash, something that has never been possible before. Furthermore, **Flash Player** 10 supports Peer to Peer (P2P) communication with Real Time Media Flow Protocol (RTMFP).
Flash Player 11	**Flash Player** 11: The major addition in this version are advanced (graphics accelerated) 3D capabilities. Other features include H.264 encoding for cameras, Native JSON support, Cubic Bézier Curves, a secure random number generator, LZMA compression for swf files as well as some other minor additions.

III. ActionScript Versions

2000–2003: **ActionScript** 1.0: With the release of Flash 5 in September 2000, the "actions" from Flash 4 were enhanced once more and named "**ActionScript**" for the first time. This was the first version of **ActionScript** with influences from JavaScript and the ECMA-262 (Third Edition) standard, supporting the said standard's object model and many of its core data types. Local variables may be declared with the var statement, and user-defined functions with parameter passing and return values can also be created. Notably, **ActionScript** could now also be typed with a text editor rather than being assembled by choosing actions from drop-down lists and dialog box controls. With the next release of its authoring tool, Flash MX, and its corresponding player, **Flash Player** 6, the language remained essentially unchanged; there were only minor changes, such as the addition of the switch statement and the "strict equality" (===) operator, which brought it closer to being ECMA-262-compliant. Two important features of **ActionScript** that distinguish it from later versions are its loose type system and its reliance on prototype-based inheritance. Loose typing refers to the ability of avariable to hold any type of data. This allows for rapid script development and is particularly well-suited for small-scale scripting projects. Prototype-based inheritance is the **ActionScript** 1.0 mechanism for code reuse and object-oriented programming. Instead of a classkeyword that defines common characteristics of a class, **ActionScript** 1.0 uses a special object that serves as a "prototype" for a class of objects. All common characteristics of a class are defined in the class's prototype object and every instance of that class contains a link to that prototype object.

2003–2006: **ActionScript** 2.0: The next major revision of the language, **ActionScript** 2.0, was introduced in September 2003 with the release of Flash MX 2004 and its corresponding player, **Flash Player** 7. In response to user demand for a language better equipped for larger and more complex applications, **ActionScript** 2.0 featured compile-time type checking and class-based syntax, such as the keywords class and extends. (While this allowed for a more structured object-oriented programming approach, the code would still be compiled to **ActionScript** 1.0 bytecode, allowing it to be used on the preceding **Flash Player** 6 as well. In other words, the class-based inheritance syntax was a layer on top of the existing prototype-based system.) With **ActionScript** 2.0, developers could constrainvariables to a specific type by adding a type annotation so that type mismatch errors could be found at compile-time. **ActionScript** 2.0 also introduced class-based inheritance syntax so that developers could create classes and interfaces, much as they would in class-based languages such as Java and C++. This version conformed partially to the ECMAScript Fourth Edition draft specification.

2006–today: **ActionScript** 3.0: In June 2006, **ActionScript** 3.0 debuted with **Adobe** Flex 2.0 and its corresponding player, **Flash Player** 9. **ActionScript** 3.0 was a fundamental restructuring of the language, so much so that it uses an entirely different virtual machine. **Flash Player** 9 contains two virtual machines, AVM1 for code written in **ActionScript** 1.0 and 2.0, and AVM2 for content written in **ActionScript** 3.0. **ActionScript** 3.0 added limited support for hardware acceleration (DirectX, OpenGL).

The update to the language introduced several new features:

Compile-time and run-time type checking—type information exists at both compile-time and runtime.

Improved performance from a class-based inheritance system separate from the prototype-based inheritance system.

Support for packages, namespaces, and regular expressions.

Compiles to an entirely new type of bytecode, incompatible with **ActionScript** 1.0 and 2.0 bytecode.

Revised **Flash Player** API, organized into packages.

Unified event handling system based on the DOM event handling standard.

Integration of ECMAScript for XML (E4X) for purposes of XML processing.

Direct access to the Flash runtime display list for complete control of what gets displayed at runtime.

Completely conforming implementation of the ECMAScript fourth edition draft specification.

Limited support for dynamic 3D objects. (X, Y, Z rotation, and texture mapping)

IV. Data Types

ActionScript primarily consists of "fundamental" or "simple" data types which are used to create other data types. These data types are very similar to Java data types. Since **ActionScript** 3 was a complete rewrite of **ActionScript** 2, the data types and their inheritances have changed.

ActionScript 2 top level data types

String - A list of characters such as "Hello World"

Number - Any Numeric value

Boolean - A simple binary storage that can only be "true" or "false".

Object - Object is the data type all complex data types inherit from. It allows for the grouping of methods, functions, parameters, and other objects.

ActionScript 2 complex data types

There are additional "complex" data types. These are more processor and memory intensive and consist of many "simple" data types. For AS2, some of these data types are:

MovieClip - An **ActionScript** creation that allows easy usage of visible objects.

TextField - A simple dynamic or input text field. Inherits the Movieclip type.

Button - A simple button with 4 frames (states): Up, Over, Down and Hit. Inherits the MovieClip type.

Date - Allows access to information about a specific point in time.

Array - Allows linear storage of data.

XML - An XML object

XMLNode - An XML node

LoadVars - A Load Variables object allows for the storing and send of HTTP POST and HTTP GET variables

ActionScript 3 primitive (prime) data types (see Data type descriptions)

Boolean - The Boolean data type has only two possible values: true and false or 1 and 0. No other values are valid.

int - The int data type is a 32-bit integer between -2,147,483,648 and 2,147,483,647.

Null - The Null data type contains only one value, null. This is the default value for the String data type and all classes that define complex data types, including the Object class.

Number - The Number data type can represent integers, unsigned integers, and floating-point numbers. The Number data type uses the 64-bit double-precision format as specified by the IEEE Standard for Binary Floating-Point Arithmetic (IEEE-754). values between -9,007,199,254,740,992 (-253) to 9,007,199,254,740,992 (253) can be stored.

String - The String data type represents a sequence of 16-bit characters. Strings are stored internally as Unicode characters, using the UTF-16 format. Previous versions of Flash used the UTF-8 format.

uint - The uint (Unsigned Integer) data type is a 32-bit unsigned integer between 0 and 4,294,967,295.

void - The void data type contains only one value, undefined. In previous versions of **ActionScript**, undefined was the default value for instances of the Object class. In **ActionScript** 3.0, the default value for Object instances is null.

ActionScript 3 some complex data types (see Data type descriptions)

Object - The Object data type is defined by the Object class. The Object class serves as the base class for all class definitions in **ActionScript**. Objects in their basic form can be used as associative arrays that contain key-value pairs, where keys are Strings and values may be any type.

Array - Contains a list of data. Though **ActionScript** 3 is a strongly typed language, the contents of an Array may be of any type and values must be cast back to their original type after retrieval. (Support for typed Arrays has recently been added with the Vector class.)

Vector - A variant of array supported only when publishing for **Flash Player** 10 or above. Vectors are typed, dense Arrays (values must be defined or null) which may be fixed-length, and are bounds-checked during retrieval. Vectors are not just more typesafe than Arrays but also perform faster.

flash.utils:Dictionary - Dictionaries are a variant of Object that may contain keys of any data type (whereas Object always uses strings for its keys).

flash.display:Sprite - A display object container without a timeline.

flash.display:MovieClip - Animated movie clip display object; Flash timeline is, by default, a MovieClip.

flash.display:Bitmap - A non-animated bitmap display object.

flash.display:Shape - A non-animated vector shape object.

flash.utils:ByteArray - Contains an array of binary byte data.

flash.text:TextField - A dynamic, optionally interactive text field object.

flash.display:SimpleButton - A simple interactive button type supporting "up", "over", and "down" states with an arbitrary hit area.

Date - A date object containing the date/time digital representation.

Error - A generic error object that allows runtime error reporting when thrown as an exception.

Function - The core class for all Flash method definitions.

RegExp - A regular expression object for strings.

flash.media:Video - A video playback object supporting direct (progressive download) or streaming (RTMP) transports. As of **Flash Player** version 9.0.115.0, the H.264/MP4 high-definition video format is also supported along side standard Flash video (FLV) content.

XML - A revised XML object based on the E4X (Standard ECMA-357); nodes and attributes are accessed differently from **ActionScript** 2.0 object (a legacy class named XMLDocument is provided for backwards compatibility).

XMLList - An array-based object for various content lookups in the XML class.