EXHIBIT 4

ActionScript

From Wikipedia, the free encyclopedia

(Redirected from Actionscript)

ActionScript is an object-oriented language originally developed by Macromedia Inc. (now owned by Adobe Systems). It is a dialect of ECMAScript (meaning it has the same syntax and semantics of the more widely known 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).[2] 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.[3]

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

ActionScript

Paradigm	Multi-paradigm: prototype-based, functional, imperative, scripting
Appeared in	1998
Designed by	Gary Grossman
Developer	Macromedia (now Adobe Systems) - Also supported in Srawl products
Stable release	3.0 (June 27, 2006; 4 years ago)
Typing discipline	strong, static
Major implementations	Adobe Flash, Adobe Flex
Influenced by	JavaScript, Java
OS	Cross-platform

ActionScript

Filename extension	.as
Internet media type	application/ecmascript[1]

Navigation

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox

What links here
Related changes
Upload file
Special pages
Permanent link
Cite this page

Print/export

Create a book
Download as PDF
Printable version

Languages

Български
Català
Česky
Deutsch
Eesti
Español

Français

Galego

Bahasa Indonesia
Italiano
עברית
Magyar
Bahasa Melayu
Nederlands
日本語
Norsk (bokmål)
Polski
Português
Română
Русский
Simple English
Suomi
Svenska

Тоҷикӣ
Türkçe
Українська
Tiếng Việt
中

behaviors and programming done in ActionScript. ActionScript 3.0 forms the foundation of the Flex 2 API.

Contents [hide]

History [edit]

ActionScript started as a 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.

Timeline by player version [edit]

- 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: Included the first version of ActionScript. Used prototype-based programming based on ECMAScript, and allowed full procedural programming and object-oriented programming.
- Flash Player 6: Added an event handling model, accessibility controls and support for switch. The first version with support for the AMF and RTMP protocols which allowed for on demand audio/video streaming.
- Flash Player 7: Additions include CSS styling for text and support for ActionScript 2.0, a programming language based on the ECMAScript 4 Netscape Proposal with class-based inheritance. However, ActionScript 2.0 can cross compile to ActionScript 1.0 byte-code, so that it can run in Flash Player 6.
- 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 (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, TR1 = LIXO 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 (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.264 video. There is a new sound API which allows for custom creation of audio in flash, something that has never been possible before.[4] Furthermore, Flash Player 10 supports Peer to Peer (P2P) communication with Real Time Media Flow Protocol (RTMFP).

Timeline by ActionScript version [edit]

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.[5] 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 a variable 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 `class` keyword 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 constrain variables 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 runtime 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)

Flash Lite [edit]

- Flash Lite 1.0: Flash Lite is the Flash technology specifically developed for mobile phones and consumer electronics devices. Supports Flash 4 ActionScript.
- Flash Lite 1.1: Flash 4 ActionScript support and additional device APIs added.
- Flash Lite 2.0 and 2.1: Added support for Flash 7 ActionScript 2.0 and some additional fscommand2 API.
- Flash Lite 3: Added support for Flash 8 ActionScript 2.0 and also FLV video playback.
- Flash Lite 4: Added support for Flash 10 ActionScript 3.0 as a browser plugin and also hardware graphics acceleration.

Syntax [edit]

ActionScript code is free form and thus may be created with whichever amount or style of whitespace that the author desires. The basic syntax is derived from ECMAScript.

ActionScript 2.0 [edit]

The following code, which works in any compliant player, creates a text field at depth 0, at position (0, 0) on the screen (measured in pixels), that is 100 pixels wide and high. Then the `text` parameter is set to the "`Hello, world`" string, and it is automatically displayed in the player:

```
createTextField("greet", 0, 0, 0, 100, 100);
greet.text = "Hello, world";
```

When writing external ActionScript 2.0 class files the above example could be written in a file named `Greeter.as` as following.

```
class com.example.Greeter extends MovieClip
{
    public function Greeter() {}
    public function onLoad() :Void
    {
        var txtHello:TextField = this.createTextField("txtHello", 0, 0, 0, 100, 100);
        txtHello.text = "Hello, world";
    }
}
```

ActionScript 3.0 [edit]

ActionScript 3.0 has a similar syntax to ActionScript 2.0 but a different set of APIs for creating objects. Compare the script below to the previous ActionScript 2.0 version:

```
var greet:TextField = new TextField();
greet.text = "Hello World";
this.addChild(greet);
```

Minimal ActionScript 3.0 programs may be somewhat larger and more complicated due to the increased separation of the programming language and the Flash IDE.

Presume the following file to be `Greeter.as`:

```
package com.example
{
    import flash.text.TextField;
    import flash.display.Sprite;

    public class Greeter extends Sprite
    {
        public function Greeter()
        {
            var txtHello:TextField = new TextField();
            txtHello.text = "Hello World";
            addChild(txtHello);
        }
    }
}
```

(*See also:* Sprite)

Finally, an example of using ActionScript when developing Flex applications, again presuming the following content to be in a file named `Greeter.as`:

```
package
{
    public class Greeter
    {
        public static function sayHello():String
        {
            var greet:String = "Hello, world!";
            return greet;
        }
    }
}
```

This code will work with the following MXML application file:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*" layout="vertical"
creationComplete="initApp()">

    <mx:Script>
        <![CDATA[
            public function initApp():void
            {
                // Prints our "Hello, world!" message into "mainTxt".
                mainTxt.text = Greeter.sayHello();
            }
        ]]>
    </mx:Script>

    <mx:Label id="title" fontSize="54" fontStyle="bold" text='"Hello, world!" Example'/>
    <mx:TextArea id="mainTxt" width="250"/>

</mx:Application>
```

Data structures [edit]

Data types [edit]

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
- Sound
- NetStream
- NetConnection
- MovieClipLoader
- EventListener

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 (-2^{53}) to 9,007,199,254,740,992 (2^{53}) 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.

Using data types [edit]

The basic syntax is:

```
var yourVariableName:YourVariableType = new YourVariableType(Param1, Param2, ..., ParamN);
```

So in order to make an empty Object:

```
var myObject:Object = new Object();
```

Some types are automatically put in place:

```
var myString:String = "Hello Wikipedia!"; // This would automatically set the variable as a string.
var myNumber:Number = 5; // This would do the same for a number.
var myObject:Object = {Param1:"Hi!", Param2:76}; //This creates an object with two variables.
// Param1 is a string with the data of "Hi!",
// and Param2 is a number with the data of 76.
```

```
var myArray:Array = [5,"Hello!",{a:5, b:7}] //This is the syntax for automatically creating an Array.
//It creates an Array with 3 variables.
//The first (0) is a number with the value of 5,
//the second (1) is a string with the value of "Hello!",
//and the third (2) is an object with {a:5, b:7}.
```

Unlike most object-oriented languages, ActionScript makes no distinction between primitive types and reference types. In ActionScript, all variables are reference types. However, objects that belong to the primitive data types, which includes Boolean, Number, int, uint, and String, have special operators that make them behave as if they were passed by value.[6]

So if a variable of a supposedly primitive type, e.g. an integer is passed to a function, altering that variable inside the function will not alter the original variable (passed by value). If a variable of another (not primitive) datatype, e.g. XML is passed to a function, altering that variable inside the function will alter the original variable as well (passed by reference).

Some data types can be assigned values with literals:

```
var item1:String="ABC";
var item2:Boolean=true;
var item3:Number=12;
var item4:Array=["a","b","c"];
var item5:XML = <node><child/></node>; //Note that the primitive XML is not quoted
```

A reference in ActionScript is a pointer to an instance of a class. This does not create a copy but accesses the same memory space. All objects in ActionScript are accessed as references instead of being copied.

```
var item1:XML=new XML("<node><child/></node>");
var item2:XML=item1;
item2.firstChild.attributes.value=13;
//item1 now equals item2 since item2 simply points to what item1 points to.
//Both are now:
//<node><child value="13"/></node>
```

Only references to an object may be removed by using the "delete" keyword. Removal of actual objects and data is done by the Flash Player garbage collector which checks for any existing references in the Flash memory space. If none are found (no other reference is made to the orphaned object), it is removed from memory. For this reason, memory management in ActionScript requires careful application development planning.

```
var item1:XML=new XML("<node><child/></node>");
delete item1;
//If no other reference to item1 is present anywhere else in the application,
//it will be removed on the garbage collector's next pass
```

ActionScript code protection [edit]

Often, Flash developers will decide that while they desire the advantages that Flash affords them in the areas of animation and interactivity, they do not wish to expose their code to the world. However, as with all intermediate language compiled code, once a .swf file is saved locally, it can be decompiled into its source code and assets. Some decompilers are capable of nearly full reconstruction of the original source file, down to the actual code that was used during creation (although results vary on a case-by-case basis).[7][8][9]

In opposition to the decompilers, ActionScript obfuscators have been introduced to solve this problem. Higher-quality obfuscators implement lexical transformations — such as identifier renaming, control flow transformation, and data abstraction transformation — that make it harder for decompilers to generate output likely to be useful to a human. Less robust obfuscators insert traps for decompilers.

References [edit]

1. ^ RFC 4329 (limit compatible with EcmaScript)
2. ^ Brimelow, Lee (2008-08-18). "Six reasons to use ActionScript 3.0". Adobe Systems Incorporated. Retrieved 2010-06-18.
3. ^ Grossman, Gary; Huang, Emmy (2006-06-27). "ActionScript 3.0 overview". Adobe Systems Incorporated. Retrieved 2010-06-18.
4. ^ "Adobe Labs - Adobe Flash Player 10.1". Labs.adobe.com. Retrieved 2009-12-17.
5. ^ Note that the name "ActionScript 1.0" is a retronym, coined after the release of ActionScript 2.0.
6. ^ "Flex 3 - Function parameters". Livedocs.adobe.com. Retrieved 2009-12-17.
7. ^ Third party review of another decompiler
8. ^ Customer comments on one Flash decompiler
9. ^ Customer comments on another Flash product

External links [edit]

- ActionScript Technology Center
- ActionScript 2.0 Language Reference
- ActionScript 3.0 Language & Component Reference
- Flex 3 LiveDocs: Programming ActionScript 3.0
- Adobe - Flash Developer Center
- Adobe Flex SDK
- Adobe's Flash Forum
- EverythingFLA online ActionScript 3.0 School

Wikibooks has a book on the topic of ***ActionScript Programming***

Wikiversity has learning materials about ***ActionScript:Introduction***

v · d · e	Adobe Flash	
File formats	.as (ActionScript) · .amf (Action Message Format) · .flv, .f4v (Flash Video) · .fxg (Flash XML Graphics) · .mxml (MXML) · .swc (Shockwave Flash Component) · .swf (Shockwave Flash)	
Software	Viewers	Adobe Flash Player · Adobe Flash Lite · Adobe Integrated Runtime (AIR) · Gnash · Lightspark · Swfdec
	Authoring Tools	Adobe Flash Professional · Adobe Flex · Adobe Flash Builder · Adobe Flash Catalyst · Adobe Flash Media Live Encoder · FlashDevelop · Open Dialect · OpenLaszlo · haXe
	Server-side software	Adobe Flash Media Server · Adobe Flash Cast
Related topics	Flash animation · Local Shared Object · Protected Streaming · Real Time Messaging Protocol · Real Time Media Flow Protocol · SWFAddress · SWFFit · SWFObject · XMLSocket	

v · d · e	ECMAScript		
Dialects	ActionScript · Caja · JavaScript (engines) · JScript · JavaScript OSA · JScript .NET · Objective-J · QtScript · WMLScript		
Engines · Comparison	Carakan · Futhark · InScript · JavaScriptCore (SquirrelFish) · JScript · KJS · Linear B · Narcissus · QtScript · Rhino · SpiderMonkey (TraceMonkey, JägerMonkey) · Tamarin · V8 · Chakra		
Frameworks	JavaScript (comparison)	Client-side (List of Ajax frameworks)	Ample SDK · Clean AJAX · CougarXML · Dojo · Echo · Ext · Google Web Toolkit · jQuery · Lively Kernel · midori · MochiKit · MooTools · OpenLink AJAX · Prototype · Pyjamas · qooxdoo · Rialto · Rico · script.aculo.us · SmartClient · SproutCore · Spry · Yahoo! UI Library
		Server-side	

& libraries		(comparison)	AppJet · Jaxer · Node.js
		Libraries (List)	PDFObject · SWFObject · SWFAddress · SWFFit
	ActionScript		PureMVC
	Multiple Implementations		Cappuccino (JavaScript / Objective-J)
People			Brendan Eich · Douglas Crockford · John Resig
Other			DHTML · Ecma International · JSAN · JSDoc · JSLint · JSON · JSSS · Sputnik · SunSpider · CommonJS

Categories: Adobe Flash | Scripting languages | JavaScript programming language family | Prototype-based programming languages