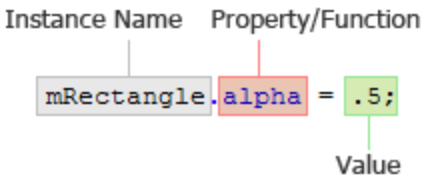
Case sensitivity

ActionScript 3.0 is a case-sensitive language. Identifiers that differ only in case are considered different identifiers. For example, the following code creates two different variables:

Dot syntax

The dot operator (.) provides a way to access the properties and methods of an object. Using dot syntax, you can refer to a class property or method by using an instance name, followed by the dot operator and name of the property or method. For example, consider the following class definition:



Literals

A *literal* is a value that appears directly in your code. The following examples are all literals:

```
17
"hello"
-3
9.4
null
undefined
true
false
```

Semicolons

You can use the semicolon character (;) to terminate a statement. Alternatively, if you omit the semicolon character, the compiler will assume that each line of code represents a single statement. Because many programmers are accustomed to using the semicolon to denote the end of a statement, your code may be easier to read if you consistently use semicolons to terminate your statements.

Parentheses

You can use parentheses (()) in three ways in ActionScript 3.0. First, you can use parentheses to change the order of operations in an expression. Operations that are grouped inside parentheses

are always executed first. For example, parentheses are used to alter the order of operations in the following code:

```
trace(2 + 3 * 4); // 14
trace((2 + 3) * 4); // 20
```

Second, you can use parentheses with the comma operator (,) to evaluate a series of expressions and return the result of the final expression, as shown in the following example:

```
var a:int = 2;
var b:int = 3;
trace((a++, b++, a+b)); // 7
```

Third, you can use parentheses to pass one or more parameters to functions or methods, as shown in the following example, which passes a String value to the `trace()` function:

```
trace("hello"); // hello
```

Comments

ActionScript 3.0 code supports two types of comments: single-line comments and multiline comments. These commenting mechanisms are similar to the commenting mechanisms in C++ and Java. The compiler will ignore text that is marked as a comment.

Single-line comments begin with two forward slash characters (`//`) and continue until the end of the line. For example, the following code contains a single-line comment:

```
var someNumber:Number = 3; // a single line comment
```

Multiline comments begin with a forward slash and asterisk (`/*`) and end with an asterisk and forward slash (`*/`).

```
/* This is multiline comment that can span
more than one line of code. */
```

Keywords and reserved words

Reserved words are words that you cannot use as identifiers in your code because the words are reserved for use by ActionScript. The compiler will report an error if you use a lexical keyword as an identifier. The following table lists ActionScript 3.0 lexical keywords.

as	break	case	catch
class	const	continue	default
delete	do	else	extends
false	finally	for	function
if	implements	import	in
instanceof	interface	internal	is
native	new	null	package
private	protected	public	return
super	switch	this	throw
to	true	try	typeof
use	var	void	while
with			

Basic function concepts

Calling functions

You call a function by using its identifier followed by the parentheses operator (`()`). You use the parentheses operator to enclose any function parameters you want to send to the function. For example, the `trace()` function is a top-level function in ActionScript 3.0:

```
trace("Use trace to help debug your script");
```

Defining your own functions

There are two ways to define a function in ActionScript 3.0: you can use a function statement or a function expression. The technique you choose depends on whether you prefer a more static or dynamic programming style. Define your functions with function statements if you prefer static, or strict mode, programming. Define your functions with function expressions if you have a specific need to do so. Function expressions are more often used in dynamic, or standard mode, programming.

Function statements

Function statements are the preferred technique for defining functions in strict mode. A function statement begins with the `function` keyword, followed by:

- The function name
- The parameters, in a comma-delimited list enclosed in parentheses
- The function body—that is, the ActionScript code to be executed when the function is called, enclosed in curly brackets

For example, the following code creates a function that defines a parameter and then calls the function using the string " `hello`" as the parameter value:

```
function traceParameter(aParam:String)
{
    trace(aParam);
}

traceParameter("hello"); // hello
```