How to emulate keyword arguments in ActionScript 3 functions

Asked 15 years, 11 months ago Modified 15 years, 11 months ago Viewed 2k times

7

Functions in Python can be called using keyword arguments of the form keyword = value. For instance, the following function:

```
def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):
    print "-- This parrot wouldn't", action,
    print "if you put", voltage, "volts through it."
    print "-- Lovely plumage, the", type
    print "-- It's", state, "!"
```

could be called in any of the following ways:

```
parrot(1000)
    parrot(action = 'VOOOOOM', voltage = 1000000)
    parrot('a thousand', state = 'pushing up the daisies')
    parrot('a million', 'bereft of life', 'jump')
```

ActionScript 3 does not have this facility. How would I best emulate it?

actionscript-3 arguments keyword

Share Improve this question Follow

asked Jan 3, 2009 at 23:03

Brett Kromkamp

4 Answers

Sorted by: Highest score (default)

7

It won't throw any errors at compile time if you mess up but you could do something like this:

```
public function awesomefunction(input:Object):void {
    if (input.foo) trace("got foo");
    if (input.bar) trace("got bar");
}
```

And then when calling it doing like this:

```
awesomefunction( { foo : 23, bar : 99 } );
```

The {} brackets create an inline object. A long winded version of the above code can be written like this:

```
var obj:Object = new Object();
obj.foo = 23;
obj.bar = 99;
awesomefunction( obj );
```

If you give it an array instead:

```
awesomefunction( [23, 99 ] );
```

([] is shorthand for an array btw)

The indices will be accesible like this:

```
public function awesomefunction(input:Object):void {
    if (input[0]) trace("got foo");
    if (input[1]) trace("got bar");
}
```

So to bring it all toghether you can use the fact that the || operator returns the value of the object that first evaluates to true, not just *true* to do:

```
public function awesomefunction(input:Object):void {
    var foo:int = input.foo || input[0];
    var bar:int = input.bar || input[1];
    if (foo) trace("got foo:", foo);
    if (bar) trace("got bar:", bar);
}
```

So, the above function will give the same output for all these three calls:

```
awesomefunction( { foo : 23, bar :55 } );
awesomefunction( { bar : 55, foo : 23 } );
awesomefunction( [ 23, 55 ] );
```

This will however not allow you to mix named and numbered, but you can do like this:

```
awesomefunction( { foo : 23, 1 : 55 } );
```

It's not very pretty, but it works!

Share

Improve this answer

Follow

edited Jan 3, 2009 at 23:53

answered Jan 3, 2009 at 23:47

grapefrukt

27k ● 6 ● 51 ● 74

4

The short answer is that in Actionscript, the order of parameters is part of a function's signature. To get around that fact, you either have to do extra work, or indulge in bad coding practices that will bite you later. I'd say you have two "good" options for getting flexibility similar to what you're looking for. The formal way would be to define a data class to act as the function's parameter. That is:

```
public class ComplaintData {
    public var state:String = "a stiff";
    public var action:String = "voom";
    public var type:String = "Norwegian Blue";
    public var voltage:String;

    public function ComplaintData( v:String ) {
        voltage = v;
    }
}

// somewhere else....

function parrot( cd:ComplaintData ) {
    trace "-- This parrot wouldn't " + cd.action;
    // etc.
}

var cd:ComplaintData = new ComplaintData( "a million" );
cd.action = "jump";
parrot( cd );
```

If the situation is more ad-hoc, a less formal approach is to make liberal use of optional parameters, and allow the caller to pass null for any non-trailing parameter they want to omit. This gets you the flexibility, but you sacrifice a fair deal of cleanliness:

```
function parrot( voltage:String,
                 state:String=null,
                 action:String=null,
                 type:String=null)  {
    if (state==null) { state = 'a stiff'; }
    if (action==null) { action = 'voom'; }
    if (type==null) { type = 'Norwegian Blue'; }
    // ....
}
parrot('1000')
parrot('1000000', null, 'jump')
```

But if both these options seem unwieldy, that's because they are. In the long run, I think it's ultimately best to internalize the idea that ordered parameter lists are part of Actionscript's grammar, and you're probably better of not trying to avoid them. Doing so either sacrifices simplicity (as above), or leads to poor practices (like using unstructured objects for parameters, as suggested in other answers, which sacrifices many benefits of type- and error-checking).

The most balanced way to work with the grammar is simply use optional parameters whenever possible, thus:

```
function parrot( voltage:String,
                 state:String='a stiff',
                 action:String='voom',
                 type:String='Norwegian Blue')  {
    // ...
}
parrot('1000');
parrot('1000000', 'pushing up the daisies');
parrot('a million', 'bereft of life', 'jump')
```

You still have to live with the fact that only trailing parameters can be omitted, but that's how the language is made (and how its native methods are defined and used).

Share Improve this answer Follow

answered Jan 6, 2009 at 4:47

fenomas

11.1k ● 2 ● 35 ● 58

+1 to using a typed object or just dealing with it. Going with ... or an anonymous object will just make your codebase more difficult to maintain. – Richard Szalay Jan 8, 2009 at 9:17

1

There is a special type in actionscript 3.0 called *...(rest)*. Put this type in the parameter list for a function and then that function is able to accept any number of arguments. The proper sytanx is: *...arrayName*, where *arrayName* is the name of the array which will hold the arguments.

```
public function forYou( ...args ) : void{
   for(var i : int = 0; i
```

Share Improve this answer Follow

answered Jan 4, 2009 at 1:04

ForYourOwnGood
40.3k ● 5 ● 31 ● 27

3 But this is just a parameter array. This does not give you named parameters, or allow you to change to order of parameter, exclude paramters, etc. – bzlm Feb 17, 2009 at 17:01

0

You could implement something similar using variable number of arguments to a function shown here.

Share Improve this answer Follow

answered Jan 3, 2009 at 23:26

Brandon
6,872 ● 11 ● 40 ● 50

Very interesting. You should add the code snippet from the site to your answer. – Soviut Jan 4, 2009 at 0:57

3 But this is just a parameter array. This does not give you named parameters, or allow you to change to order of parameter, exclude parameters, etc. – bzlm Feb 17, 2009 at 17:02