

## Tutorial – Using Sound in ActionScript 3

Flash Tutorial March 24th, 2008

This tutorial is Part 1 of a series of tutorials on creating a music player.

Anyways, in an era where melodic snobbishness is so prevalent, everyone with a website (or a MySpace account) wants to play their favorites for the whole world to hear, so if you're a Flash designer/developer, it might do you some good to learn how to import, play, and modify sound in ActionScript. In this tutorial, I'm going to introduce you to using sound in ActionScript 3, and in later posts, we'll start to get a little more in depth with the different things we can do with sound.

Note: If you're easily bored with tutorials and just want to look at some ActionScript, be sure to check out my recent post on an ActionScript 3 / XML Music Player I created. This post contains a link to an FLA that might have some of the ActionScript you're looking for.

In this tutorial, we're not going to talk about importing a sound into Flash. Rather, we're going to discuss accessing external sound files using ActionScript without ever actually importing them into the Flash authoring environment. 1. Save your Flash file. Before we do anything else, we need to save our Flash file, because we'll soon be pointing to an external sound file, and in order to point to the right location for the sound file, Flash will need to know where the FLA file will be saved as well. In this example, I'll be saving my FLA file in the same directory as my mp3 music file. 2. Create a Sound Object. So far, all we're trying to do is to get a sound file to play, so we don't really need to worry about putting anything on the stage. Instead, simply select "Layer 1" on your timeline, rename the layer "Actions", click on frame 1 of this layer, and hit Option+F9 (or just F9 on PC) to open up your Actions panel for this frame. The code for creating a new sound object is simple:



To create a sound object, we first created a variable called "music" and we set this variable equal to a new Sound object. Inside the parentheses for this new Sound object, we placed a new URLRequest object, which allows us to access the URL of an external file, as we see in line 1. This external file is the mp3 file we will be playing. In line 2, we point to our new sound object and tell it to play. And with only two lines of code, we're already playing music in our Flash file!

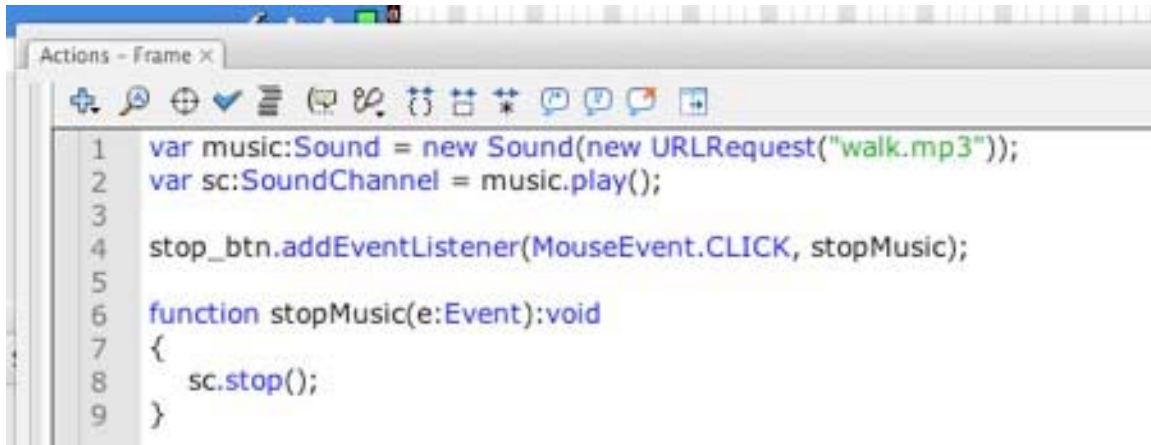
Phase Two: Shut that Racket Off!

If I were you, I wouldn't even play the music by default unless you're building a website for a band, or some other website where the user might expect to hear default music. But one thing's for sure--if you set up your music to play by default, you WILL lose some traffic to your site. Even if you provide controls to turn the music off, many people aren't going to take the time to look for the stop button. They're just going to leave. Create a stop button. It doesn't matter what the stop

button looks like, as long as it looks like a stop button. So create a new layer, draw your graphics, and then convert to a button symbol by hitting F8. Here's what my stop button looks like (in true, glossified, Web Two-Dot-Oh fashion):

## Stop button

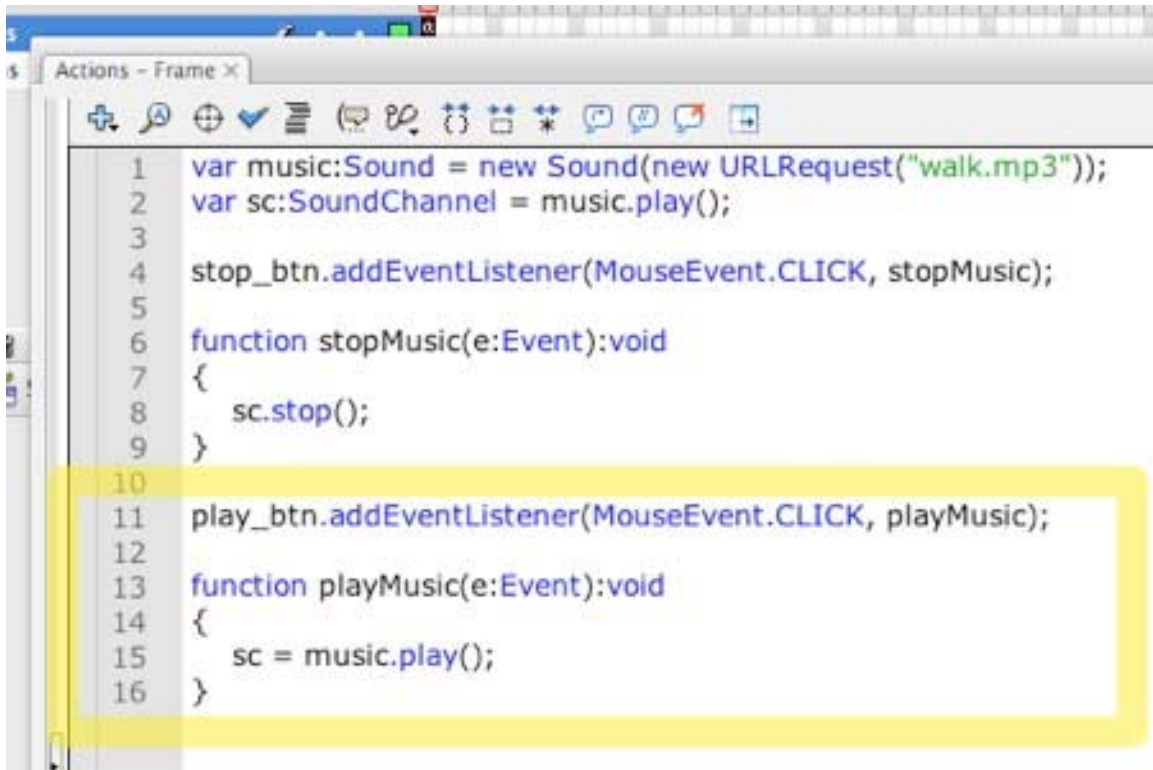
I've also given my button an Instance Name of `stop_btn`. Here's the code that causes our stop button to work (hang on to your seats, this one's complicated):



The first thing you should notice is that line 2 has changed a little. Unlike with ActionScript 2, in AS3, you can't control the playback, volume, etc. of a sound object using the Sound class. Instead, you have to assign the playback of your sound to a new instance of the SoundChannel class. And then, using your new SoundChannel object (which we simply called "sc"), you can tell your sound how to behave. In line 4, we created the typical event listener for our button so that when we CLICK on our stop button, it will trigger the "stopMusic" function, which contains the code for stopping the playback of our music. On line 8, inside the aforementioned "stopMusic" function, notice that the "stop()" method is attached to the SoundChannel object we created, which we called "sc." And now, if we test our file, we'll see that we have the ability to stop our sound.

So, How About a 'Play' Button?

Now that we've stopped the sound, how do start it back up again? EASY! Create your play button (I'm giving mine an Instance Name of "play\_btn") and add the following code to your Actions layer:



```
1  var music:Sound = new Sound(new URLRequest("walk.mp3"));
2  var sc:SoundChannel = music.play();
3
4  stop_btn.addEventListener(MouseEvent.CLICK, stopMusic);
5
6  function stopMusic(e:Event):void
7  {
8      sc.stop();
9  }
10
11  play_btn.addEventListener(MouseEvent.CLICK, playMusic);
12
13  function playMusic(e:Event):void
14  {
15      sc = music.play();
16  }
```

At least that's the code you'd THINK you might add! And if your user presses the right buttons at the right time, everything will be dandy! But try testing your movie and pressing the play button after the movie has already started playing, and you'll discover that you've got the same sound playing on top of itself. So before we tell our music to start playing again, we need to make sure it isn't already playing. And we're going to do this by creating a variable to keep track of our status. Let's call our variable "isPlaying" and set it to a data type of Boolean (true/false). Here's what your code will look like:

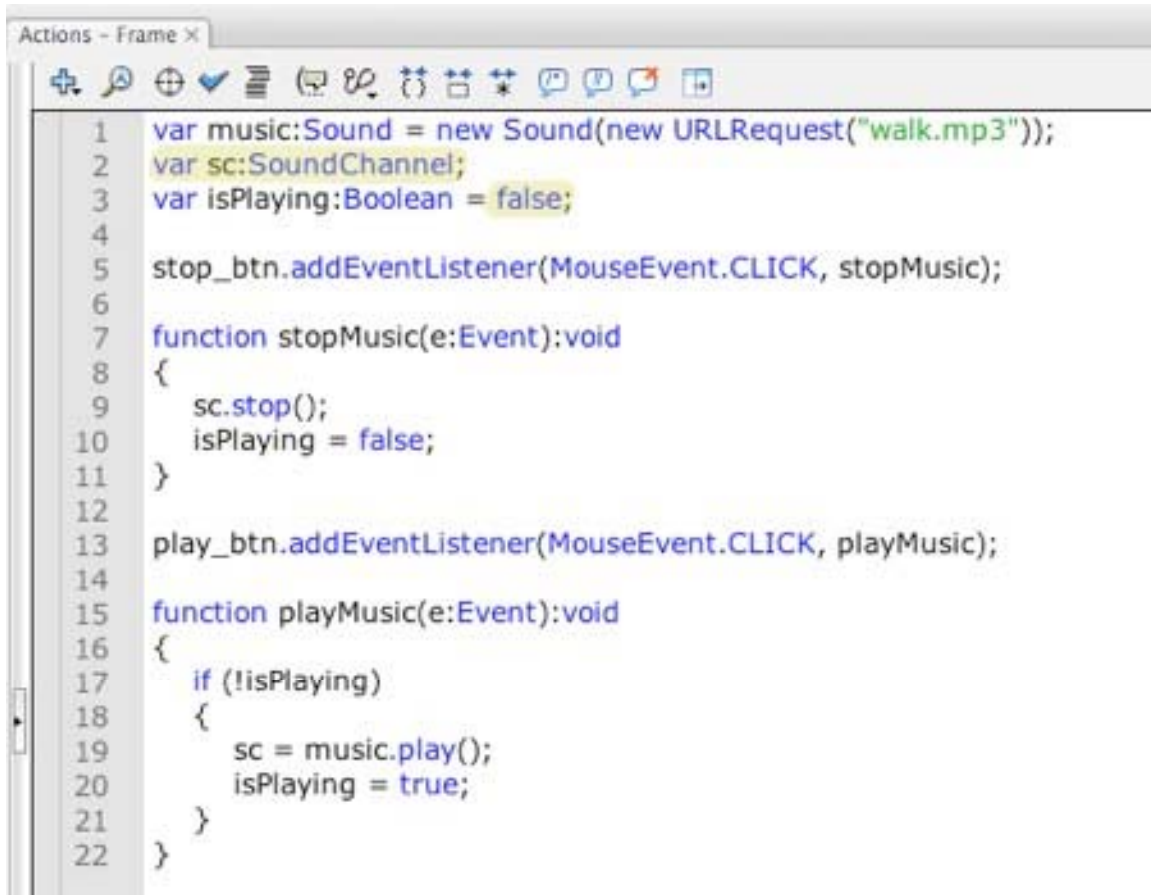
A screenshot of an IDE window titled "Actions - Frame x". The window contains a toolbar with various icons for editing and testing. Below the toolbar, there is a list of 22 lines of ActionScript code. The code is as follows:

```
1 var music:Sound = new Sound(new URLRequest("walk.mp3"));
2 var sc:SoundChannel = music.play();
3 var isPlaying:Boolean = true;
4
5 stop_btn.addEventListener(MouseEvent.CLICK, stopMusic);
6
7 function stopMusic(e:Event):void
8 {
9     sc.stop();
10    isPlaying = false;
11 }
12
13 play_btn.addEventListener(MouseEvent.CLICK, playMusic);
14
15 function playMusic(e:Event):void
16 {
17     if (!isPlaying)
18     {
19         sc = music.play();
20         isPlaying = true;
21     }
22 }
```

On line 3, we created our variable and set it equal to "true" since our music is playing by default. Then, inside the stopMusic function, we set our variable to false. Inside the playMusic function, instead of just allowing the music to start playing any time we hit the play button, we included an "if" statement that first checks to see if the music is already playing, and if it's NOT playing already (the exclamation point basically means "not"), then we allow it to play, and we set our "isPlaying" variable back to true. Now our music player functions beautifully!

### One Last Thing

Remember earlier, when I mentioned that it's generally a bad idea to have your music playing by default when the page loads? Yeah, well I meant it! So let's talk about how we can set this up. Actually, you've probably already figured this part out for yourself. Remember that "music.play()" statement in line 2? Get rid of it! Wait until the user actually clicks on the play button before you start playing the music. Of course that also means you need to set the "isPlaying" variable to "false" in line 3. Here's your final code:

A screenshot of an IDE window titled "Actions - Frame x". The window contains a toolbar with various icons for editing and testing. Below the toolbar is a code editor with 22 lines of ActionScript code. The code defines a Sound object, a SoundChannel, and a Boolean variable, then sets up event listeners for two buttons to play and stop the music.

```
1 var music:Sound = new Sound(new URLRequest("walk.mp3"));
2 var sc:SoundChannel;
3 var isPlaying:Boolean = false;
4
5 stop_btn.addEventListener(MouseEvent.CLICK, stopMusic);
6
7 function stopMusic(e:Event):void
8 {
9     sc.stop();
10    isPlaying = false;
11 }
12
13 play_btn.addEventListener(MouseEvent.CLICK, playMusic);
14
15 function playMusic(e:Event):void
16 {
17     if (!isPlaying)
18     {
19         sc = music.play();
20         isPlaying = true;
21     }
22 }
```

Notice that in line 2, we still declared our "sc" variable and strict-typed it to SoundChannel, even though we didn't set it equal to anything. The reason for this is that if we try to declare our "sc" variable inside the "playMusic" function, then that variable won't be accessible outside of the function. By declaring the variable outside of the function, we're ensuring that it will be available anywhere within our code. Anyways, those are the basics of creating a simple music player.