

# Flash Button & Movie Symbol

## Introduction

ActionScript is a very powerful language—we barely scratched the surface of what it's capable of in the last lesson. In today's lesson, the plan is to make the scratch a bit deeper. It's one thing to control how an animation plays back using Frame actions, but doing so only lets you map out the playback. The user can't control it by making choices while it's happening.

It's all about interactivity—adding the ability to let users make choices about where they want to go in your Flash movies. That's when you'll need to know how to use a couple more Flash tools. So far, you've only used the graphic symbol, but today we'll go over the button and movie clip symbols, and I'll show you what they can do to help with interactivity.

When I was learning Flash, this is the point in my training when I started having the most fun. And while Flash can be challenging, it's fair to say that it gets more exciting from this point on.

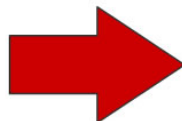
Before we begin, please save files to your H:/Drive by going to the G:/Drive > Flash > Assignment > Button, you will need these files for this lesson.

## Buttons

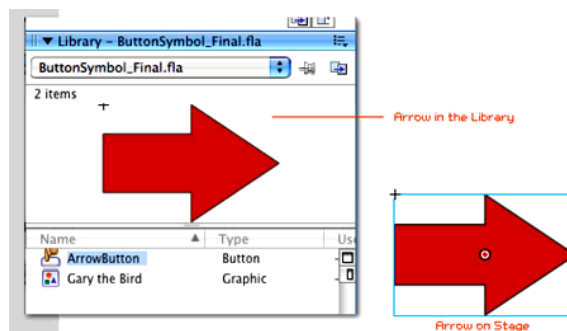
The Flash button symbol can be a very useful tool. In fact, I don't think I make any Flash content lately that doesn't use at least one button. So let me show you how to make your very own button from scratch.

A button in Flash doesn't have to conform to the typical idea of a round object that appears to stick out from the page. You can literally make any shape or creation into a button symbol, just as you've done in the past with graphic symbols. Just keep this in mind as we go through this example.

1. Start by opening *ButtonSymbol.fla* from the lesson files. The Stage and Timeline are empty, but in the Library you'll find our old friend *Gary the Bird*. We'll use him later on.
2. Save this file as a new file named *MyButtonSymbol.fla* so we can make changes.
3. Create a red arrow on the Stage. This will become your button, so don't make it too huge. There are a few different ways to make the arrow, but I just used the **Pen** tool and created points.

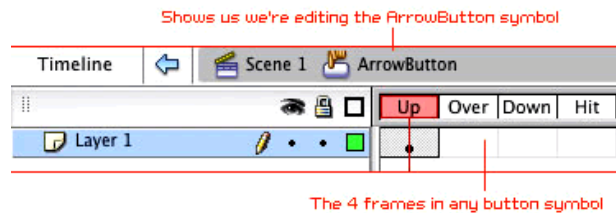


4. Use the **Selection** tool to select the entire arrow by either double-clicking its fill or by drawing a selection rectangle around it. Then press F8 to Convert to Symbol.
5. When the Convert to Symbol window appears, name the new symbol *ArrowButton* and choose **Button** as the type. Finally, click **OK** to create the new button symbol.



As you can see, the result is very similar to when you created graphic symbols in past lessons. The icon next to the button in the Library is a little different, though.

Double-click the *ArrowButton* symbol in the Library. This will take you into Edit Mode, and this is where things get a little different. As you can see, the *ArrowButton*'s Timeline isn't the same as other Timelines we've seen. Instead, there are only four frames, and they have labels. Hmmm, what could it all mean?



Each of these frames has a special purpose that affects how the button will function. When you convert anything to a button symbol (as we did), the artwork is always placed on the first frame (the *Up* frame).

To put it quite simply, the *Up* frame represents what the button looks like when it's just sitting there on the Stage. Right now, the arrow shape is on the *Up* frame, and this is what the button will look like. But if you look around on various Web sites, you'll see buttons that change appearance slightly when you interact with them. That's where these other frames come into play.

1. Keep the *Over* frame selected (not the label above it) and press F6 to Insert a Keyframe. As we've talked about before, doing this will copy the contents of the previous keyframe (the *Up* frame, in this case) to the new one.
2. The *Over* frame of a button represents what the button will look like when a user moves the mouse over the button. You don't have to make it different from the *Up* state, but we're going to.

Before we do, let's see what the movie looks like right now. Press CTRL + ENTER or COMMAND + ENTER to test your movie. When you roll over the arrow, you can see that it looks the same, but the mouse cursor changes, telling you that it's a clickable button.

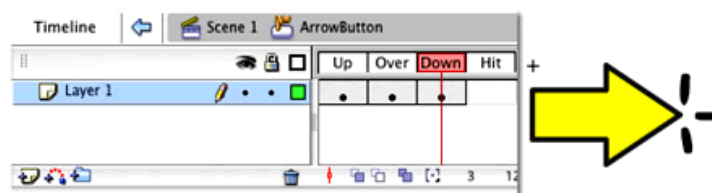
3. Change the fill color of the arrow shape on the *Over* frame to yellow.
4. Test the movie again to see the difference.

Before you close the Test Movie window, click the arrow button to see that nothing happens.

5. After returning to the button Timeline, repeat the steps above to create a keyframe on the *Down* frame. This makes a copy of the yellow arrow that you can work with.

The *Down* frame defines how a button will look when the user moves the mouse over the button and holds down the mouse button. In other words, it's how the button looks when you click it.

6. Select the entire stroke of the arrow shape by double-clicking the outline. In the Property inspector, set the **Stroke** size to 4 instead of 1.
7. Use the **Brush** tool to draw three lines near the tip of the arrow. This makes it look like its being clicked.



8. Finally, test your movie—now you can roll over and click the button to see all that we've done come to life.

9. Save your work by going to **File > Save**.

You did it! You created a fairly decent-looking interactive arrow button in just a few short steps.

I'm sure you're wondering about one thing, though. What about the *Hit* frame you saw in the button Timeline? What does that do?

It's rare that you'll use the Hit frame at first, but here's what it's for: The *Hit* frame defines the size and shape of the clickable area of the button. Right now, you have an empty Hit state. This means that the user can only click the exact shape of the button to make the action begin. In this case, it's what you've drawn on the Down frame. By the way, when you're in Test Movie Mode, try moving your mouse ever so slightly to the right over the squiggles and notice that this area is in the clickable Hit area, too.

## Movie Clips

Perhaps the most powerful and widely used symbol in Flash today, the *movie clip* symbol is very similar to the graphic symbol, but with some pretty big exceptions.

Every symbol in Flash has its own individual Timeline. Sure, the button Timeline is a bit different (with only four frames), but it's still a Timeline. These Timelines can support the same animations, symbols, text, and other stuff that the main Flash Timeline can. And that means you can put an animation inside a symbol and then animate that symbol.

For example, check out our old friend Gary here. As he walks across the field, a fly is buzzing around his head.

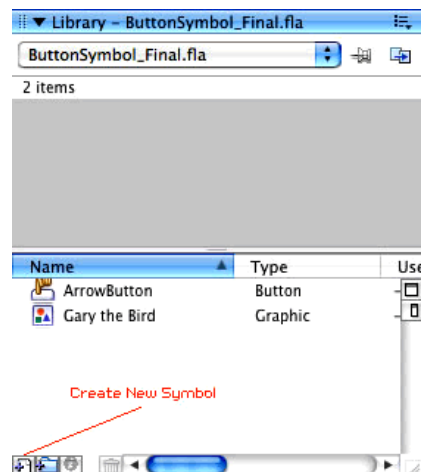
The fly is a graphic symbol that's animating inside a movie clip and that movie clip is inside the Gary graphic symbol. Then we're animating Gary's graphic symbol on the main Timeline using a motion tween. Sound confusing? Well, sure it is when you say it all at once like that.

As we move forward, I'd like you to think of graphic symbols as containers for nonmoving shapes or artwork, and think of movie clip symbols as containers specifically for animations. It's not a good idea to cling to this rule forever, but it's a good place to start while we go over the differences.

Let's go back to the file we worked on before called *MyButtonSymbol.fla* and learn a little about movie clip symbols.

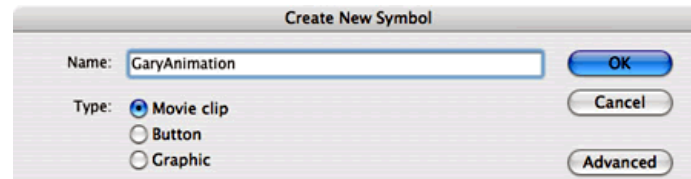
In the past, we've created symbols by selecting some existing artwork and pressing F8 to *Convert to Symbol*. This time, we'll create a brand-new empty symbol and then add artwork to it.

Look at the Library panel of the *MyButtonSymbol.fla* file, and you'll see that your *ArrowSymbol* and Gary the bird are already there. Let's add a new symbol by clicking the **New Symbol** button at the bottom-left corner of the Library.

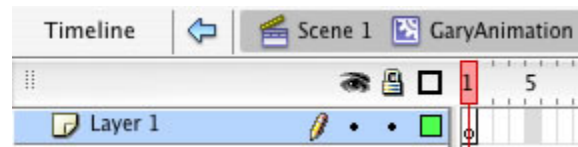


This brings up the same new symbol window, but this time it says *Create New Symbol* at the top instead of *Convert to Symbol*.

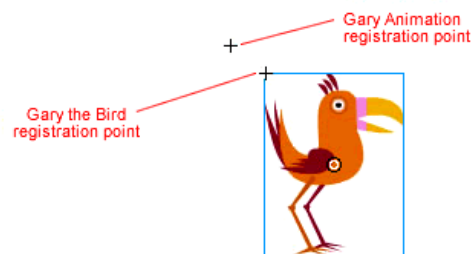
You're going to create a movie clip symbol that contains an animation of Gary, so set the **Type** to **Movie clip** and name the symbol *GaryAnimation*. Click **OK** to create the symbol.



Because you've created an empty symbol, Flash takes you to the new symbol's Timeline so you can create some artwork inside it. This is a bit different from when you convert something into a symbol.



Now it's time to animate Gary, so let's start by dragging him (the Gary the Bird symbol) from the Library onto the Stage.



Take a look at the registration points in the figure above. A symbol's registration point will come in handy in the near future, and we'll talk more about it then. But for now, just know that the registration point for any symbol is shown by crosshairs as you see above, and it's Flash's way of letting you know where the symbol's top left corner or it's 0,0 x and y coordinate is.

It really doesn't mean all that much right now, but try to put Gary as close to that top-left registration point as you can.

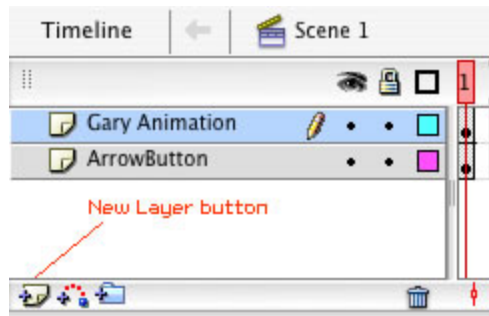
Now rename the layer in the Timeline to something that's more informative (this is just good practice). Let's name the layer *Gary*.

We want to animate Gary walking across the Stage, so create a keyframe on frame 20 so we can add a tween for him to move. Select frame 20 and press F6 to insert a new keyframe.

Wait, we've run into a little snag here. Right now, you're editing the movie clip symbol in Edit mode, which means you can't see the main Stage. Why is that important? Well, what if you move Gary too far to the right and he ends up going off the Stage? You won't know until you put him in place, and then you have to make changes. Not a big deal, but let's prevent the extra work.

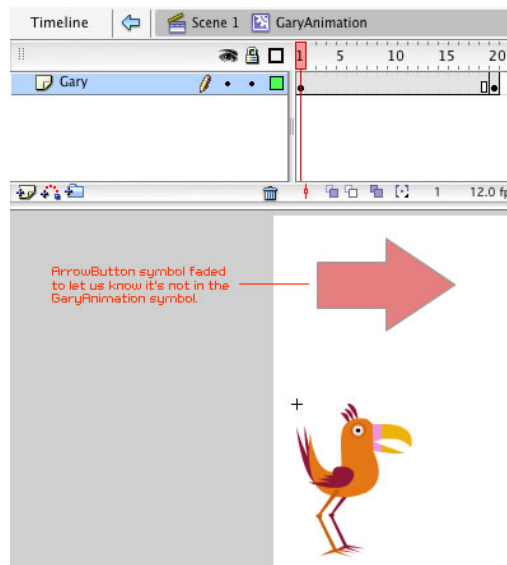
What you want to do is go back to the main Timeline and place Gary's animation there. Then you can edit the movie clip *in place* so you can see just how far to move him.

Jump back to the main Timeline by clicking **Scene 1** at the top. Gary disappears because you haven't put him on the main Stage yet. Before you do that, let's create a new layer for him and name it *Gary Animation*.

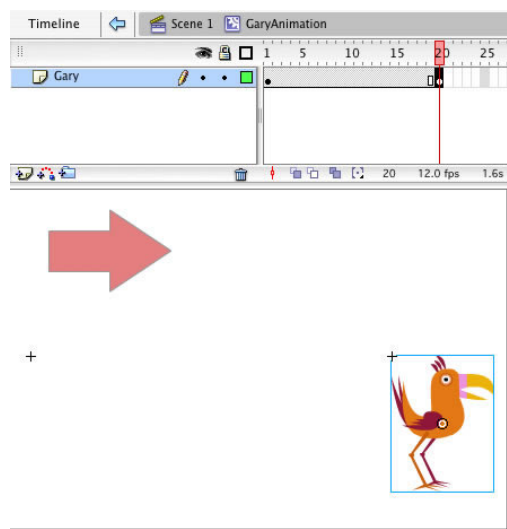


Select the *Gary Animation* layer in the Timeline and then drag the *GaryAnimation* movie clip from the Library onto the Stage. Place him at the lower-left corner of the Stage so he has room to move.

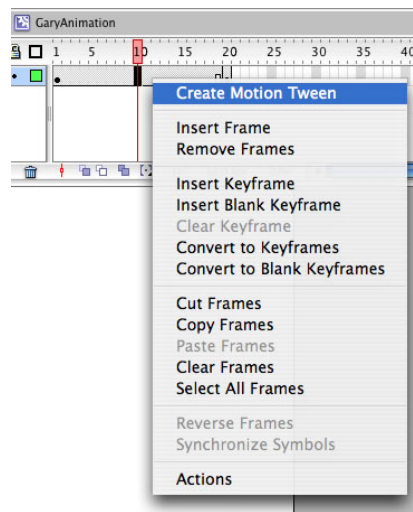
Now when you double-click the GaryAnimation symbol instance on the Stage, you go into *Edit in Place* mode, which lets you see what you're doing in context.



Once inside the *GaryAnimation* symbol, move the playhead to the keyframe on frame 20. To make Gary move, you'll have to grab him on the Stage and move him to the right. So go ahead and move him to the right side of the Stage now.



Finally, add a motion tween to the frames between 1 and 20 so Flash will move Gary across the Stage. To do this quickly, just right-click a frame between the keyframes and choose **Create Motion Tween** from the list.



The animation's in place, so let's return to the main Timeline by clicking **Scene 1** and see what you've done. Go ahead and test your movie, and you should see something similar to this:

When you close the test window and return to the main Timeline, you'll notice that there's only one frame on the main Timeline, but the movie clip you created still plays. This is one of the cool features of movie clips. Their Timelines run completely independent of the main Timeline.

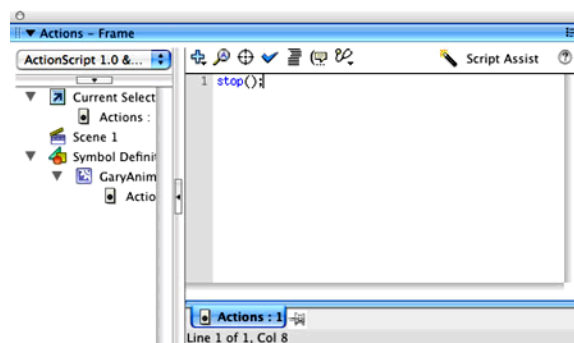
Save your file before you move on to the next chapter, but don't close it. We're going to make our button and our movie clip talk!

## Buttons and Movie Clips Join Forces

Right now, the movie clip (*GaryAnimation*) is sitting on a single frame on the Timeline, and yet it still loops when you test the movie. What if you wanted to control the animation by preventing it from looping? If you put a stop frame action on the main Timeline, it will only keep that Timeline from playing, but the movie clip has its own Timeline. So what do you do?

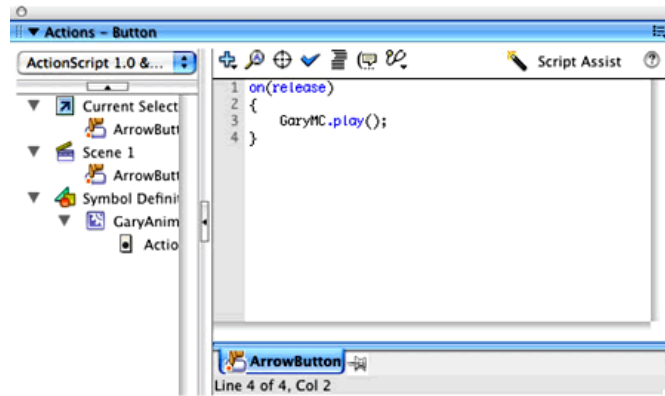
Here's where it gets interesting. We're going to make it so the movie clip doesn't play until you click the ArrowButton.

1. Double-click the *GaryAnimation* symbol instance that's on the Stage.
2. Now that you're working on the symbol's Timeline, you can control its playback. Create a new layer and name it *Actions*.
3. Select the first keyframe on the Actions layer and open the Actions panel. Type `stop();` into the script pane. This will prevent the *GaryAnimation* Timeline from playing automatically.



1. Good, now we need to set up the button to trigger the animation. Go back to the main Timeline by clicking Scene 1.
2. Select the ArrowButton instance on the Stage and look back to the Actions panel. You'll see it now says *Actions—Button* at the top. This tells you that any action you type in now will be added to the button you have selected. An action placed on a button symbol is appropriately called a *button action*.
3. The key difference between button actions and frame actions is that a button action requires what's called an *event handler*. Don't worry about the technical term—it's just an extra bit of text that you'll need to add.

So with the *ArrowButton* selected, type in the script pane everything you see in the figure below:



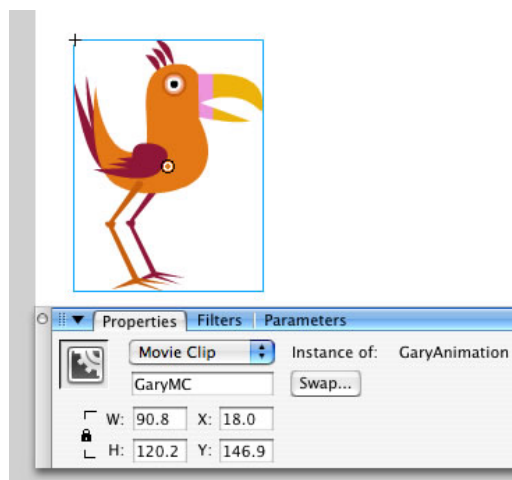
1. I know—that's a lot of new stuff, so let's break it down. Everything between the first bracket { and the second bracket } are the actions. The `on(release)` part is the event handler that I mentioned. It's basically saying, "When someone clicks on this button and then *releases* the click, perform the following actions." Then the actions are between the brackets. Right now, we only have one action, and let's look at it.

```
GaryMC.play();
```

This action is saying, "Start playing the GaryMC Timeline."

I know what you're thinking: "What's the GaryMC Timeline?" *GaryMC* is actually the name of the movie clip instance on the Stage. The problem is, we haven't named the symbol instance yet. So let's do that now.

2. Select the instance of the *GaryAnimation* symbol on the Stage and look at the Property inspector. You'll see a field that says `<Instance Name>`. Type in *GaryMC* as you see below.



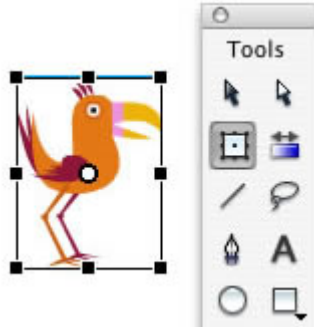
1. Why did I name it *GaryMC* and not *GaryAnimation* to match the symbol in the Library? Well, I could've named it that, but I like to choose names that are a little different from one another so I don't get confused.

Now the actions that you added to the button can find the movie clip you want to control because you've named it.

2. Try it out by running the Test Movie command.

The frame action on Gary's Timeline prevents him from moving, but then when you press the button and release, you tell his Timeline to play. Pretty cool, huh? Now it's time to play.

1. Select the *GaryMC* instance and use the **Free Transform** tool to make it smaller.

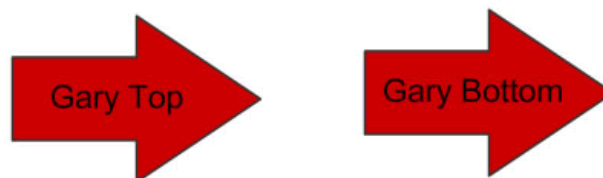


Using the Free Transform tool to adjust the scale of a symbol instance

2. Now make a copy of the *GaryMC* instance by selecting it and pressing CTRL + D or COMMAND + D on Mac.
3. Move that new instance just above the original. Right now, it has the same instance name because you made a copy. This will cause problems if you don't change it. So name the top one *GaryTopMC* and the bottom one *GaryBottomMC*.
4. Select the *ArrowButton* instance again and change the action to `GaryTopMC.play()` ;
5. Test your movie and, when you press the button, only the top movie clip animates. If you go back and change the button action to `GaryBottomMC.play()` ; then only that movie clip instance will respond to the button.

At this point, you can even duplicate the ArrowButton and put a different action on the copy so that you have one button to control each instance of the movie clip. So let's do that.

1. Select the ArrowButton instance on the Stage and press CTRL + D or COMMAND + D on Mac to make a copy.
2. With the Actions panel open, select each button and change one's action to `GaryTopMC.play()` ; and the other to `GaryBottomMC.play()` ;
3. Now activate the **Text** tool and put some text inside each arrow so you know which is which.



Two instances of the *ArrowButton* symbol with text to label them

When you test your movie now, you should end up with something like what you see here.

Even with this very basic setup, you can start to see the fun that's possible when you get deeper into Flash. I'm just enjoying making the two Garys race each other by clicking the buttons one after the other.

As you can see, the animations play completely independent of one another.

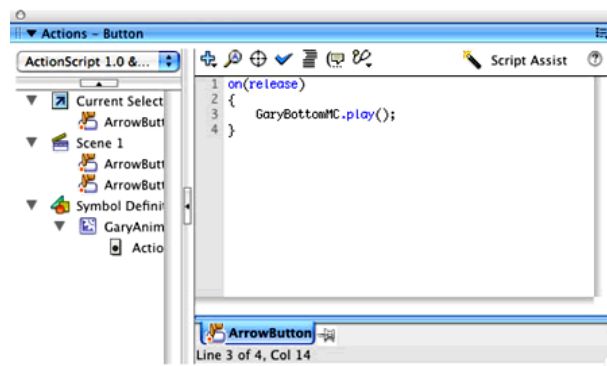
## Other Actions

Right now, you're only telling Flash to play the Timeline when you click a button, but you could easily change those play actions to the stop or goto actions we used in the last lesson.

Let's give it a try. Change the action on one of the arrow buttons to `GaryTopMC.play()`; and the other to `GaryTopMC.stop()`; When you test your movie, you'll see that you can start and stop the animation by clicking the right button.

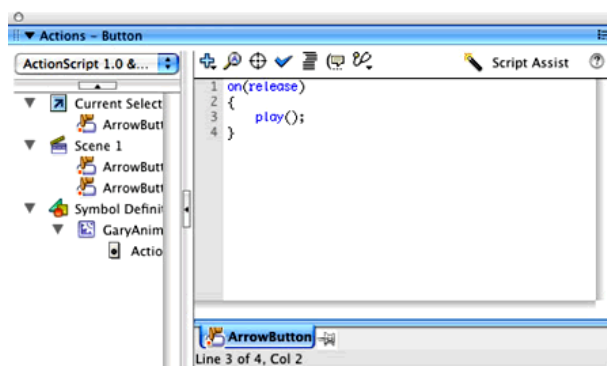
Of course, you still have that stop frame action in the GaryAnimation Timeline, so that means it'll stop automatically anyway, but if you're quick or you remove that frame action, you can gain more control.

Instead of this:



Button action for controlling a movie clip instance's Timeline

You'd just have this:



Button action for controlling the Timeline that the button is sitting on

Flash assumes that you're trying to control the Timeline that the buttons are on unless you tell it otherwise.

The final part of the button actions to mention is the event. The event is the word that's inside the parentheses of the button action. In the example, we used the `release` event when we said `on(release)`, but there are other events you can use. Here are the main ones to know:

**press**—This will execute the button action immediately when you click the button instead of waiting for the release.

**rollOver**—This executes the action when you roll over the button. No clicking required.

**rollOut**—This executes the action when you roll over the button and then roll off of it.

Feel free to replace `release` with any of these to have some fun. Be sure to type them exactly as they appear above or they won't work.

## Assignment

In this assignment, you're going to recreate the functionality of a DVD player by assigning the proper actions to buttons and keyframes of an existing Flash document. Open *Button\_Assignment.fla* from the Assignment Folder.

Inside, you'll find three buttons: *Play*, *Pause*, and *Stop*. Below the buttons, you'll find a movie clip instance with an actual movie inside. I chose to put an actual movie clip in there so you could get a sneak preview of what we'll be talking about in later lessons. Don't worry—we'll talk more about that soon.

Right now, the movie plays automatically and loops over and over. Your job is to add button actions and frame actions where they're needed so the movie doesn't start until you press the Play button and then you can control the playback using the other buttons.

The Stop button should act like a DVD player's stop button. This means that when you press Stop, the movie should stop playing and go back to the beginning. Save as *period\_lastname\_button\_assignment.swf*.

Good luck and have fun!

## Project

Return to your Superhero silent film. Create three of your own buttons: *Play*, *Pause*, and *Stop*. The buttons should allow the view to control the animated movie. Your job is to add button actions and frame actions where they are needed so the movie is controlled by the viewer. Save as *period\_lastname\_superhero\_button.swf*

## Turn-in

1. My Button Symbol as *period\_lastname\_mybuttonsymbol.swf*
2. Button Assignment as *period\_lastname\_button\_assignment.swf*
3. Superhero Button as *period\_lastname\_superhero\_button.swf*

## Quiz

In a Word Document type your answers the following questions:

1. What Frame on the button's Timeline defines what the button looks like when it's sitting there?
2. What kind of action should you use to prevent your movie from looping as it does by default?
3. If you were to put `on(release) { stop(); }` on a button instance, what would happen if the user clicked that button during playback?
4. What must you place a frame action on?
5. What is the purpose of putting a frame action on its own layer?