Movie Structure: Keyframes, Labels and Callback Functions

One of the most prominent features of Flash is its use of layers. Like Macromedia Fireworks, Adobe Photoshop, and other graphics programs, layers can be used to isolate and group content. Layers are placed on top of each other, which allows content in a higher layer to cover content in a lower layer.

Unlike these other programs, layers in Flash exist on a timeline. This means that an object can move on its layer as the timeline moves from frame to frame. ActionScript code can be added to frames and can be assigned to execute when the player reaches a particular frame.

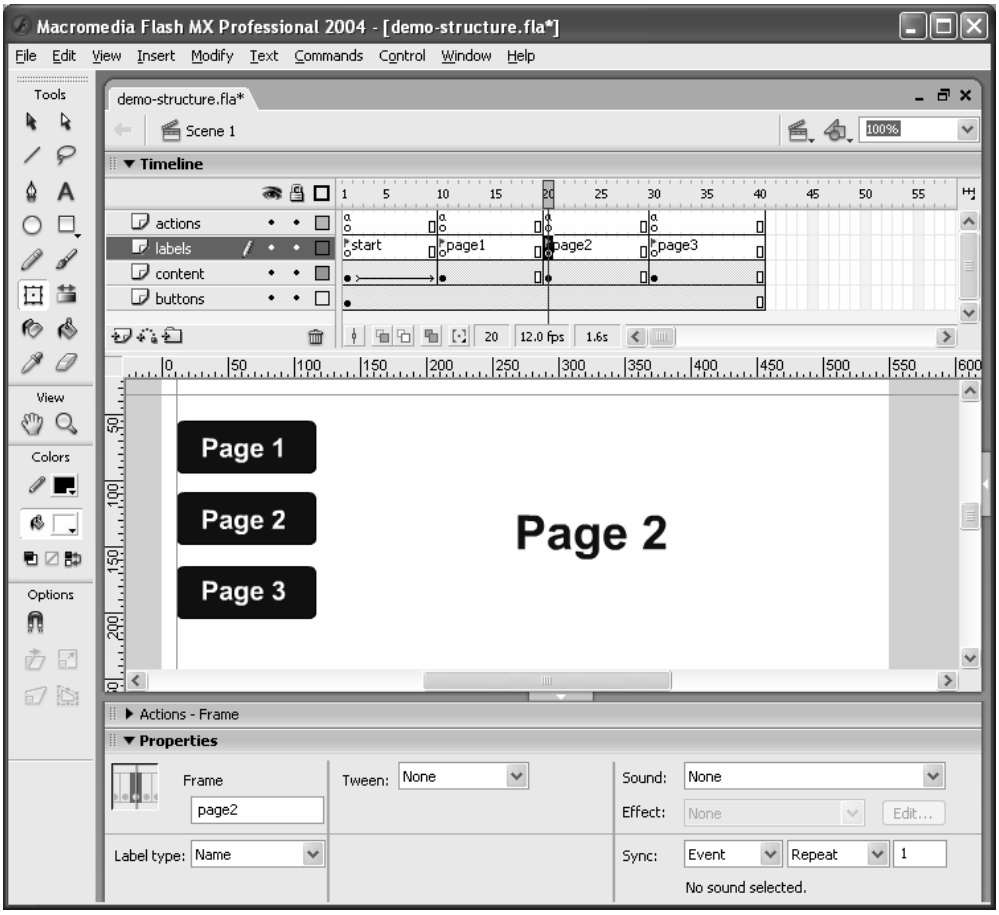
Movie Structure

It is important to structure a Flash movie in a way that makes sense to you and the others who must edit it. Two important elements of a well structured Flash document are organization and consistency. If you follow a set of short rules or conventions, you can be sure that everyone on your team will understand your logic. (Also, you will be sure to understand your document when you revisit it months after it is created!) Here is short list of suggestions:

1. Clearly name each layer.
2. Create a layer named “labels” that will hold nothing but frame labels.
3. Create a layer named “actions” that will hold nothing but timeline actions.
4. Keep these two layers on the top of the list so that they will be easy to find.

Demo: Movie Structure

The following demo, saved as **demos/demo-structure.fla**, shows a simple movie that follows these rules.



Deeper Examination: demo-structure.fla

The demo above is very simple. There are three "pages" which have different content. The three buttons on the left allow the user to navigate between the pages.

Testing Movies

If you would like to test this movie yourself, open **demos/demo-structure.fla** in Flash and press **F12** (to open in a browser) or **Control-Enter** (to test the movie right in Flash). Flash will automatically create the needed .swf or .html files.

Movie Structure

The demo above has four layers. Two layers hold visible content: The layer named "buttons" holds the three blue buttons on the left while the layer named "content" holds the text in the center of the page that says "Page 1," "Page 2," or "Page 3." Additionally, the layer named "actions" holds only timeline actions and the layer named "labels" holds only frame labels

By selecting frame 20 on the "content" layer you will see the same thing that appears in the screenshot above.

Labels Layer

Any *keyframe* may be labeled. Labels provide an easier method for identifying a section of the movie than always referring to the frame number. This demo has four labels: start, page1, page2 and page3. Labels are displayed two ways in the timeline. They appear on the labeled frame as small red flags. If there are enough frames to the right of the flag, the actual label will also appear in the timeline.

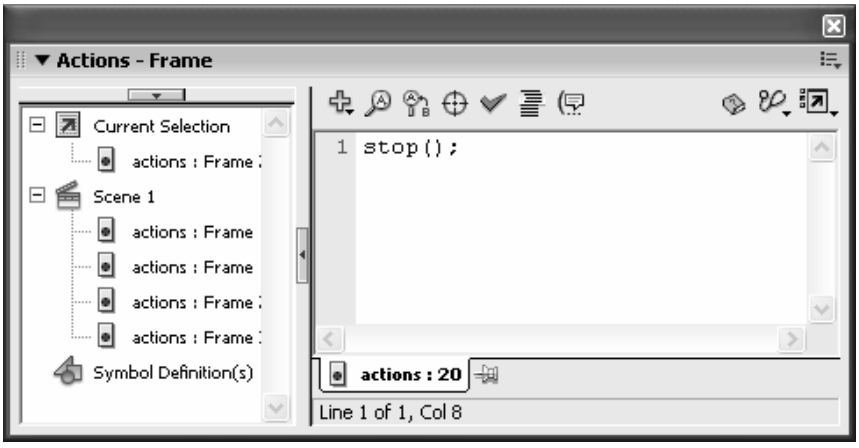
Technically, any keyframe can be named. However, while each keyframe *may* be named, there should never be two labels within the same frame, but on different layers. This could cause confusion both for Flash and for humans examining the document! If all labels are stored on one layer named "labels" they will always be easy to find.

Actions Layer

While actions can exist in several places throughout a movie, it is recommended that they only be placed on the "actions" layer. This will make actions easy to find. Also, if you accidentally place actions in multiple layers within the same frame, only one will work.

Synchronous Actions (Timeline Actions)

Some actions are triggered by time. When the movie reaches a certain frame, the action will execute. For example, there is a simple action that is repeated in Frames 10, 20 and 30. It is the **stop()** action. The screenshot below displays the Actions Panel from Frame 20 for this demo, **demos/demo-structure.fla**.



Asynchronous Actions (Event-driven Actions)

Other actions are triggered by an event that may occur at any time. For example, in this movie the user may click on any of the three buttons at any point in time. Some users may click on certain buttons again and again, while some buttons may never be used.

The code for this type of action may also appear in the timeline, but it is not triggered by time. It is triggered by the assigned event. Both of these types of actions will be used frequently in this course.

Event-driven Actions (sometimes called Button Actions), are a bit more complicated. They will be covered in detail later, but here is a n example of the buttons in the demo above.

Review of Library Items: Buttons, Movie Clips and Graphic Symbols

The Library is arguably one of the most important features within Flash. It allows for reuse of page elements (thus helping reduce file size) and is required in "tweening" (or automatically moving an item be*tween* one place and another). Here is a quick review of the different types of symbols:

1. **Graphic Symbols** – Graphic Symbols store any graphic (including text) that will be reused throughout a movie. These symbols must be used in "motion tweening" and help reduce the overall movie file size. There is often little or no action in a graphic symbol because the timeline of a graphic symbol is not independent like a Movie Clip Symbol.

2. **Movie Clip Symbols** – The timeline in Movie Clip Symbols plays independently from the main movie. This is critical because they may be started and stopped on demand. They may even loop continuously while the main movie is stopped. Also, since Movie Clips are accessible via the Movie Clip Hierarchy (as will be covered later in this course) they can be manipulated with ActionScript.

3. **Button Symbols** – Button Symbols allow for interactivity that users expect on the web. They may change when the user rolls over them or clicks on them. They may also have actions assigned to them that will affect another part of the movie.

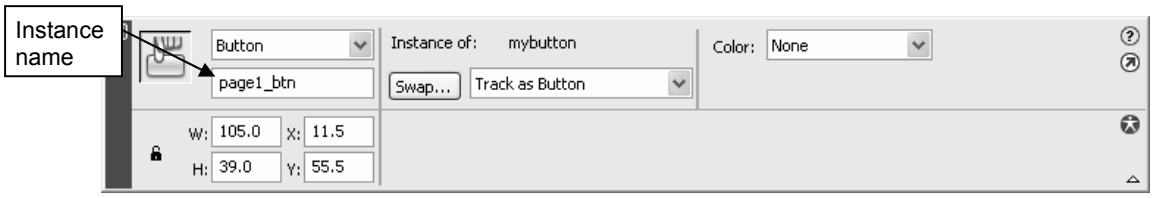
Remember, since symbols may be placed in symbols, it is possible for a movie clip to hold another movie clip which has graphic symbols and buttons within it! As movies get more and more complicated, it can be difficult to follow someone else's design. This is why consistent organization is so important!

The Library from our demo, **demos/demo-structure.fla**, appears below. It shows one button (named "mybutton") and one graphic symbol (named "page1text").

Basic Actions – Adding callback functions to buttons

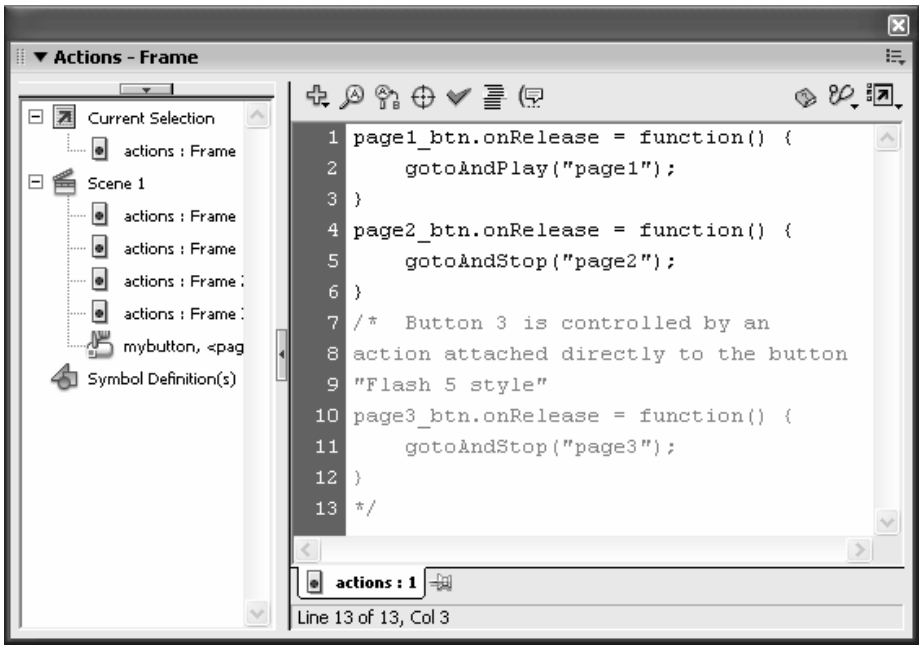
As you have already seen, our demo, **demos/demo-structure.fla**, has three buttons on the left side of the main movie. These three buttons are actually three instances of the same button symbol that are all on the "buttons" layer. Text is placed on top of the instances to show what they do. For example, the button that says "Page1" will take the user to that page.

Before an action can be attached to an instance of a button, the instance must be named using the properties panel. This button is named "page1_btn."



The ActionScript code that controls the first two buttons is located in the Actions Panel in Frame 1 of the actions layer. The name of the button appears and is followed by an event. Flash is told to watch for this particular event, onRelease, which occurs when the user clicks (and releases) the button. The code to be executed appears within curly braces and following the word "function()." This is called a callback function.

```
page1_btn.onRelease = function() {
   gotoAndPlay("page1");
}
```



Note about Flash 5 syntax on buttons

Did you notice that the code above appears in Frame 1 in the timeline? If you are familiar with Flash 5, you may find the above syntax to be strange. Before Flash MX was released, the only way to attach actions to buttons was to select the button itself and enter the desired action in the Actions panel. This is called a button action (or object action). In fact, the third button in this demo is created in this way. It appears below:

```
on(release){
   gotoAndStop("page3");
}
```

It is still possible to code buttons in this manner, but since small pieces of code are scattered throughout the document, it is less desirable than using callback functions. In order to edit code using this style, you must actually select each button. To edit code written using callback function style, the code can generally be held within the same frame.