Getting a handle on ActionScript
A basic primer for non-programmers
ActionScript (AS)

- Similar to JavaScript
- Reusable pieces of code
- Action panel in Flash automates process
- Keyboard shortcut F9 opens AS panel
- Flash has code snippets which really help
- Most code is placed on the Main Timeline
A class is a blueprint, or concept, of something. Compare the movie clip to button symbols. What are some differences between the two?

Movie clips have nearly an unlimited number of frames, and they play when your Flash movie is running unless you use ActionScript to tell them otherwise.

Buttons have a Timeline with only four frames and do not play unless you roll over or click them.

The MovieClip and Button classes are the blueprints for all movie clips and buttons respectively. Though each symbol may look different, all members of each class have certain similarities; properties and methods.
 Instantiateion occurs when you drag a symbol to the stage from the Library panel in Flash.

One symbol can have many instances

You give instance names to movie clips, buttons, and text fields but not graphic symbols.

To give an instance name to an instance, select it and type the name in the Instance Name field in the Property Inspector.
Instances

- In ActionScript, you refer to instances by their instance names, not by their symbol names in the Library panel.

- Always start with a lowercase letter, and do not use spaces or special characters other than underscores.

- It is a best practice to pick a naming convention and use it consistently. Some examples of good instance names are btnContact, mcNav, and grSnowflake.
■ You can create a **Linkage** to a symbol in your Library which eliminates the need for the object to be placed on the stage.

■ To create Linkage click on the **Export for ActionScripting** checkbox in the Advanced section of the Symbol Properties dialog box.

■ This allows you to communicate with that symbol with ActionScripting.
Properties

- Comparable to ADJECTIVES - How an Instance appears.
- Each named object can have its own values set to its properties.
  - There are hundreds of properties depending on the **Class** of object including: height, width, rotation and many others.
  - **Ex:** `mcCat.scaleX += .5;`  
    *Increases size of the movie clip instance 50%*
- You can define and change the properties of each object or instance through ActionScript.
Methods

- Comparable to VERBS - things objects can do
- When an object does something using a method we say the method is called or that the object calls the method
- Example; `mcCat.play();`
  
  *Calls the play method and makes the movie clip mcCat play its timeline.*
Properties and Dot Syntax

- ActionScript uses dot syntax to put together objects and properties and assign a value

- OBJECT. PROPERTY = VALUE

- movieclip1.rotation = 45
  - We use multiple dots to maintain object hierarchy. For example if we wanted to control the width of a symbol (mcBall) nested inside another symbol (mcBallSet) the syntax would be:

- mcBallSet.mcBall.scaleX = 1.5;
Methods and Dot Syntax

- Methods are called in the same way
  - Example; `mcBall.gotoAndPlay ("start");`

- The parenthesis after `gotoAndPlay` signifies a method rather than a property

- The statement inside the parenthesis is an argument or a parameter. The example above goes to the `start` frame label placed on the timeline of `mcBall`.

- Semicolon indicates the end of a sentence
  - Example: `stopAllSounds();`
Variables

- Variables are **containers** for data types

- Comparison with Flash CS6 structure
  - **Symbols** are containers for different objects you create
  - **Variables** are containers you create with ActionScript
  - They store the values of the different objects you create
  - They can change dynamically
Variables in detail

Think of a game where the player has a score. The data about that player’s score is contained in a variable. When the player gets more points, the number in the score variable increases. Thus, the score variable acts as a container (or variable) for a number (or data).

In ActionScript 3.0, you create variables using the keyword var. The code to create a variable called score is `var score`.

Variables can hold many types of data; text values, such as a user name, password, or text in a text field. They can also hold true or false values, such as whether or a user logged into a Web site has administrator status.

The type of data a variable holds is called its data type. In ActionScript 3.0, you must give variables a data type. To tell Flash the type of data a variable will hold type in a colon and then the data type. Most data types begin with a capital letter. Example var score:Number;
Functions

- Functions hold multiple actions

- The primary purpose of functions in AS is to have them available to be called as needed.

- Advantages of Functions:
  - Flexible
  - Convenient
  - Reusable
  - Centralized

- Created in one place but executed from anywhere in the movie.
  - Ex.- three buttons in three MCs with the same purpose. We can put one block of code in a function on the main timeline and invoke it from each button as needed.
Functions Vocabulary

- Declaration - creating the function
- Invocation - calling the function
- Arguments and parameters - providing the data to manipulate
We need a function name and a block of statement to perform
function functionName () {
    Statement one;
    Statement two;
    Statement three;
}

- Curly braces begin and end a function block
- Creating the function does NOT execute the function
Most common way to invoke the function is to target a named object, add an event listener and call the function.

```javascript
home_btn.addEventListener(MouseEvent.CLICK, home);
```

In this case the function’s name is “home”
Parameters are the variables that can be used within the function

We provide a list of identifiers between the parenthesis of the function declaration

```javascript
function moveClip (theClip, xDist, yDist) {
    theClip.x += xDist;
    theClip.y += yDist;
}
```

We replace hard-coded values = more flexibility

- Allows you to modify how all the actions within that function behave
Events are things that happen while a Flash movie is playing. Many types of events exist, such as when a visitor to your Web site clicks a button, presses a key on the keyboard, or starts downloading a file. You can utilize events by running functions when events happen. The special functions that run when events happen are called event handlers.
To write an event handler, create an event handler function. It receives information about the event that makes the function run.

The code to create an event handler function call which reacts to a button click and plays the timeline is:

```actionscript
function playMovie(event:MouseEvent) :void{
    play();
}
```

The `event :MouseEvent` code in the parenthesis is how you capture information about what caused the function to run.

The event part represents the event that happens and the colon specifies the data type of this event, which is MouseEvent.
To attach an event to an event handler, use an event listener. Event listeners wait for events to happen, and invokes the appropriate event handler function.

To invoke an event handler function use the addEventListener method. You type the instance name, type a dot, and type addEventListener. Then in parentheses, specify the type of event the instance is listening for, type a comma, and type the name of the function.

For example, if you had a button with an instance name of btnPlay and you wanted to run a function called playMovie whenever you clicked it, you would type the following:

```
btnPlay.addEventListener(MouseEvent.CLICK, playMovie);
```