CS 116 – Action Script MovieClip

Elie Abi Chahine

Introduction

- The MovieClip class is the core class for animation and movie clip symbols created in Adobe Flash.

- It has all the behaviors and functionality of display objects, but with additional properties and methods for controlling a movie clip's timeline.

- Movie clips are a key element for people who create animated content with the Flash authoring tool and want to control that content with ActionScript.

- Whenever you create a movie clip symbol in Flash, Flash adds the symbol to the library of that Flash document. By default, this symbol becomes an instance of the MovieClip class, and as such has the properties and methods of the MovieClip class.

MovieClip Class

• Like I said, the MovieClip class is the core class for animation and movie clip symbols created in Adobe Flash.

•The Movie Clip Class contains a lot of properties like:

- x , y (variables representing the object's position on the screen)
- scaleX , scaleY (variables representing the objects scale values)
- rotation (variable representing the object's 2D rotation value)

• Those are just a few, we will cover a lot more as we go…

MovieClip Class

Let’s create our first Movie Clip:

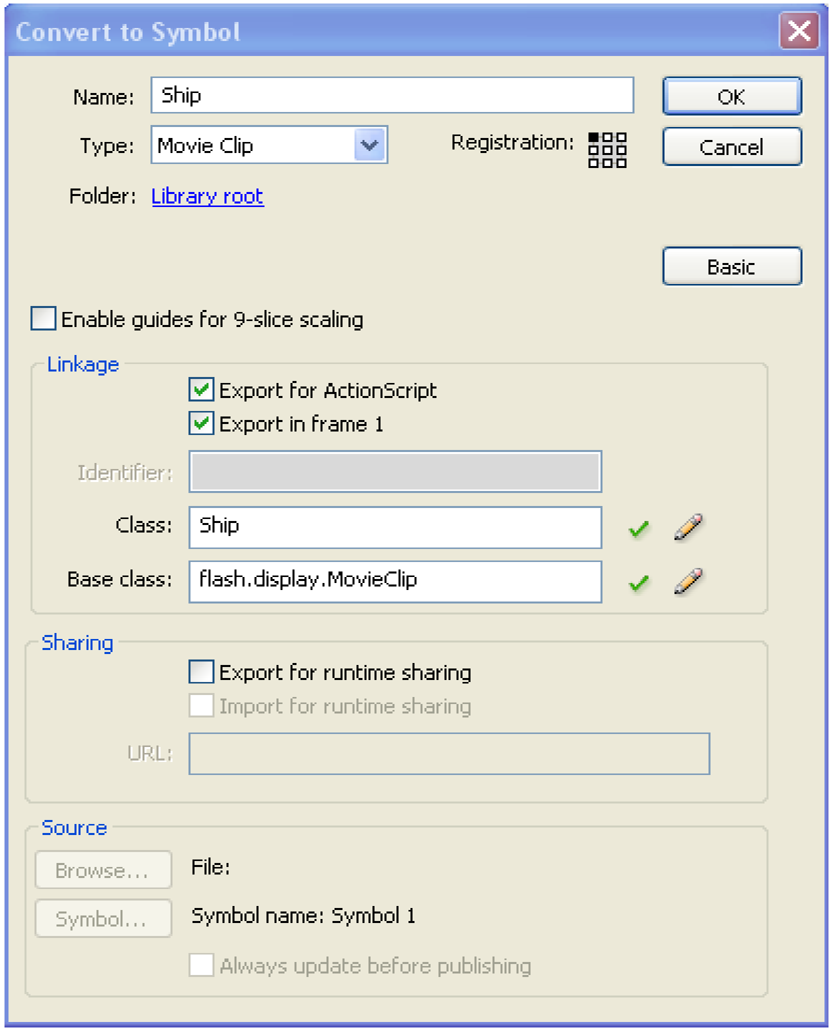
• Using the “Brush Tool”, draw your main character (don’t forget that we want him to move around on the screen so don’t make him too big)

• Once you’re done, use the “Selection Tool” and select your character.

•Right Click and “Convert to Symbol”

Exporting MovieClips in Flash

- The convert to symbol dialog box should be opened now.
- You should name your symbol
- Set the Type to "Movie Clip"
- Under linkage do the following:
  - Check "Export for ActionScript"
  - Name the Class (usually the same name as above)
  - Notice that the "Base Class" is "flash.display.MovieClip" which means that our symbol is linked to the MovieClip Class.
- Press on "OK"



Convert to Symbol

Name: Ship

Type: Movie Clip

Registration:

Folder: Library root

Enable guides for 9-slice scaling [ ]

Linkage

[✓] Export for ActionScript

[✓] Export in frame 1

Identifier:

Class: Ship

Base class: flash.display.MovieClip

Sharing

[ ] Export for runtime sharing

[ ] Import for runtime sharing

URL:

Source

File:

Symbol name: Symbol 1

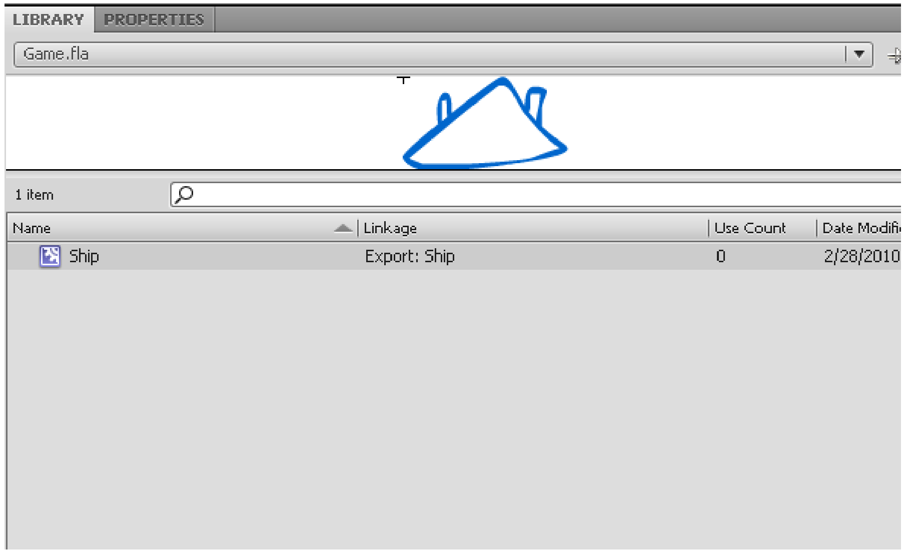
[ ] Always update before publishing

Exporting MovieClips in Flash

• Now check the editor's top right, under "Library" you will see your new symbol added there

•On stage, you will see your original object. Select it and delete it.

•Notice that it is still there in the Library, which proves that the one on stage was just a copy.



Name	Linkage	Use Count	Date Modifi
Ship	Export: Ship	0	2/28/2010

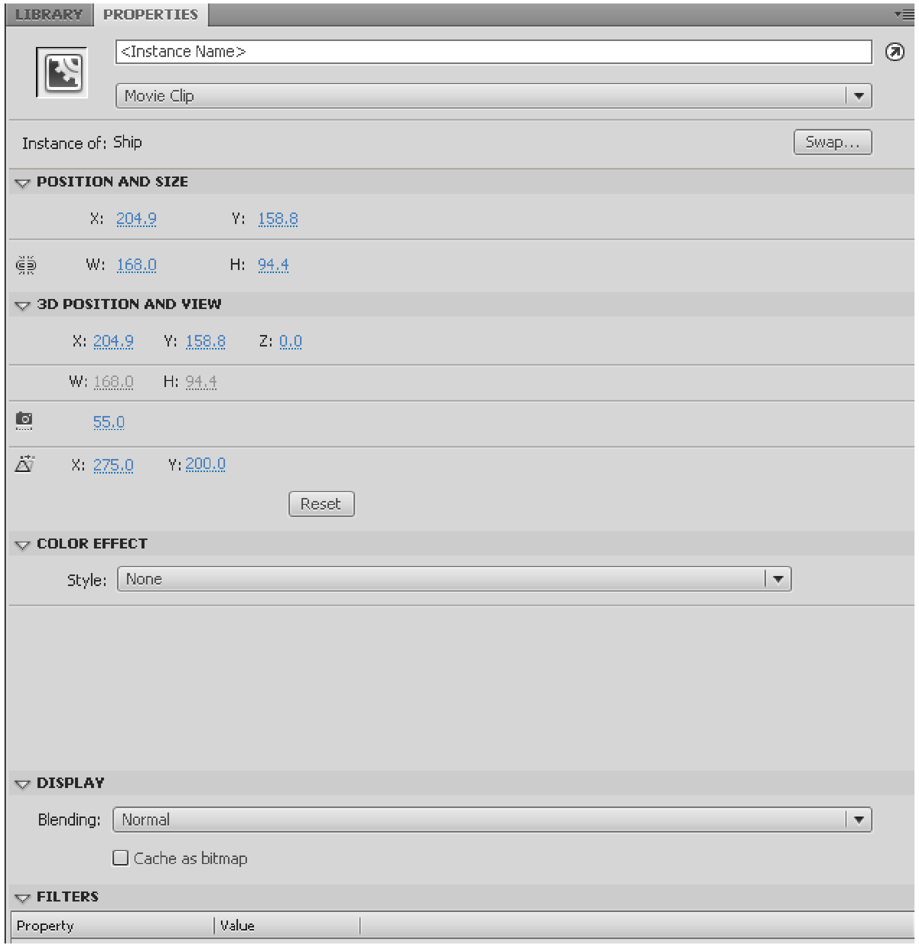
Creating Static Instances

- Go to the Library, click on the symbol's image, drag and drop a copy on the stage.

- Under Properties, you will find <Instance Name>

- Click on it and name the copy that you just created.

- Once you named it, it means you just created your 1st MovieClip Class object



Instance of: Ship

X: 204.9 Y: 158.8

W: 168.0 H: 94.4

X: 204.9 Y: 158.8 Z: 0.0

W: 168.0 H: 94.4

55.0

X: 275.0 Y: 200.0

Style: None

Blending: Normal

☐ Cache as bitmap

Creating Static Instances

•Run the code. You should see the object's copy you just put on stage

• Go to the stage's "Actions" window and write the following code:

```
mcShip.x = 0;
mcShip.y = 0;
```

<u>**NB:**</u> "**mcShip" should be replaced with whatever you called your object.**

• Run the code again. Now the object should be on the screen's top left corner. Basically, since our object is linked to the MovieClip class, then we have access to it's properties.

Creating Static Instances

• This method of creating objects is usually used when you are creating a defined number of objects from a type (although I'm not a big fan of it).

• What if I want to shoot bullets from my ship?? I need a way to dynamically create objects….

• Of course when I say "dynamically create", it means it will be at runtime and using code.

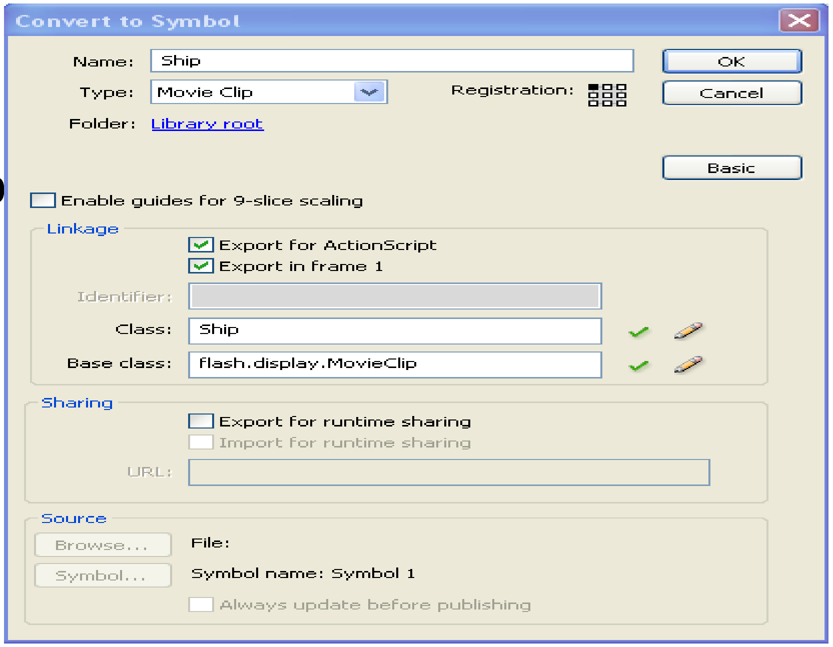
• Let's see how we can do that, but first delete all the instances you have on the screen (stage).

Dynamically Creating Instances

- When converting to symbol and exporting to ActionScript, we specified a class name and specified that the base class is the MovieClip class.

- That means we created a new class that inherits from the MovieClip class. When I say inherits, it means it will take all the properties found in the base class and then adds it's own.

- In this image on the right, we are creating a Ship class that inherits from the MovieClip class. So the Ship class contains all the attributes found in the MovieClip class and later we will learn how to add more attributes to it.


Convert to Symbol

Name: Ship
Type: Movie Clip
Registration:
Folder: Library root

[ ] Enable guides for 9-slice scaling

Linkage
[✓] Export for ActionScript
[✓] Export in frame 1
Identifier:
Class: Ship
Base class: flash.display.MovieClip

Sharing
[ ] Export for runtime sharing
[ ] Import for runtime sharing
URL:

Source
File:
Symbol name: Symbol 1
[ ] Always update before publishing

Dynamically Creating Instances

How to dynamically create an instance from the Ship class?

Since it is a class (object) then we will do the same thing as if we are creating any other instance from another class (Array, String…)

```
var Name:Type = new TypeConstructor();
```

Example:

```
var mcShip:Ship = new Ship();
```

Note: If you run the following code, the ship won't show on the screen (stage). You will need to add the object you created as a child to the stage in order to be rendered.

```
stage.addChild(mcShip);
```

Rendering a MovieClip

- The “stage” will handle all the rendering. That is why everything that you need to show on the screen has to be added as a child to the stage.

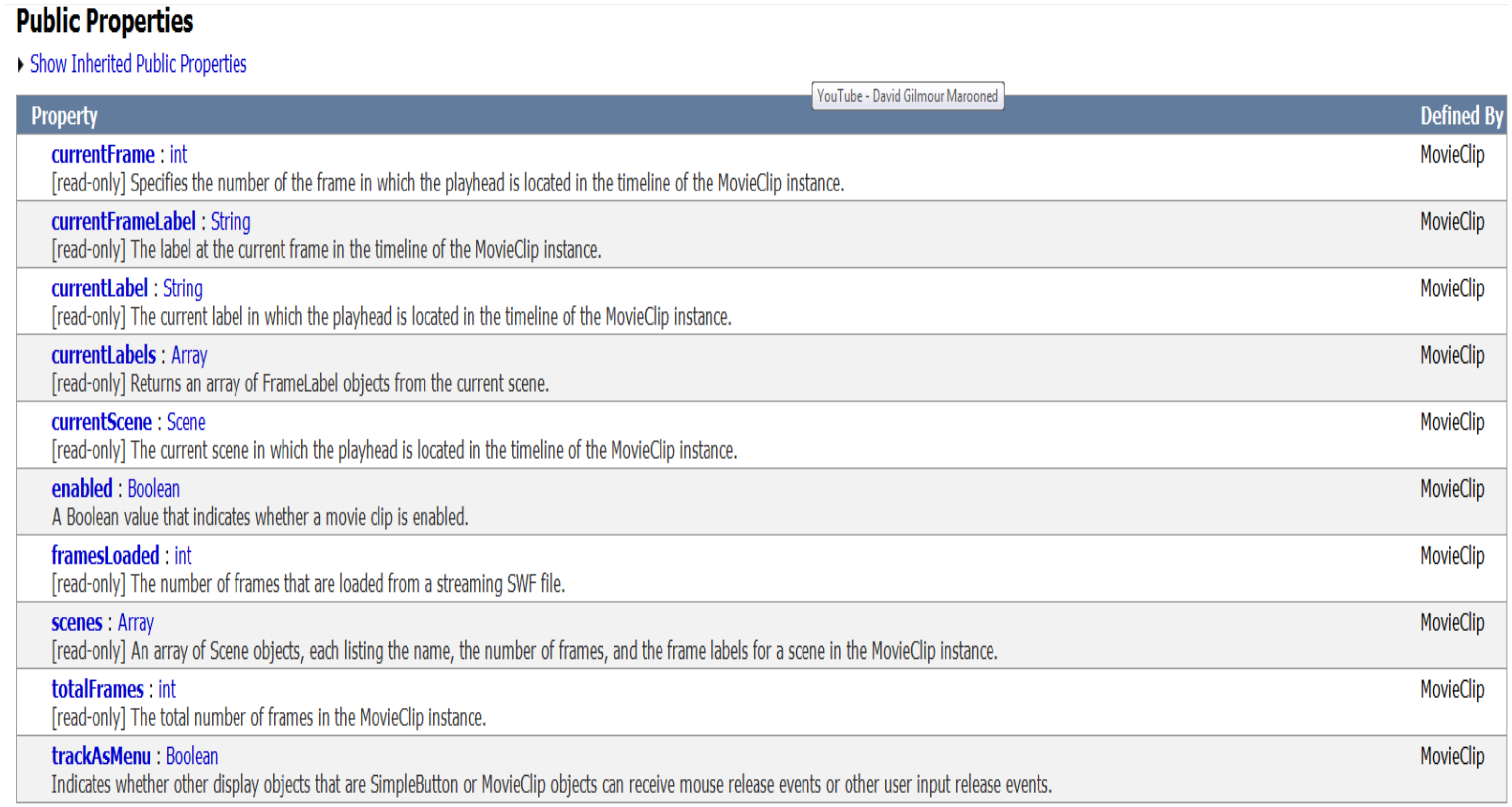
- The order you add them will be the order they get rendered. First added first rendered.

- So their order in the stage’s list will specify their Z Order. That’s how we specify/manage which object will be drawn on top of the others.

- You can re-add any object (even if it was added before) to reposition him on top of the rendering list.

Attributes

- Since the MovieClip class is part of ActionScript, then we can check it's attributes in the help.



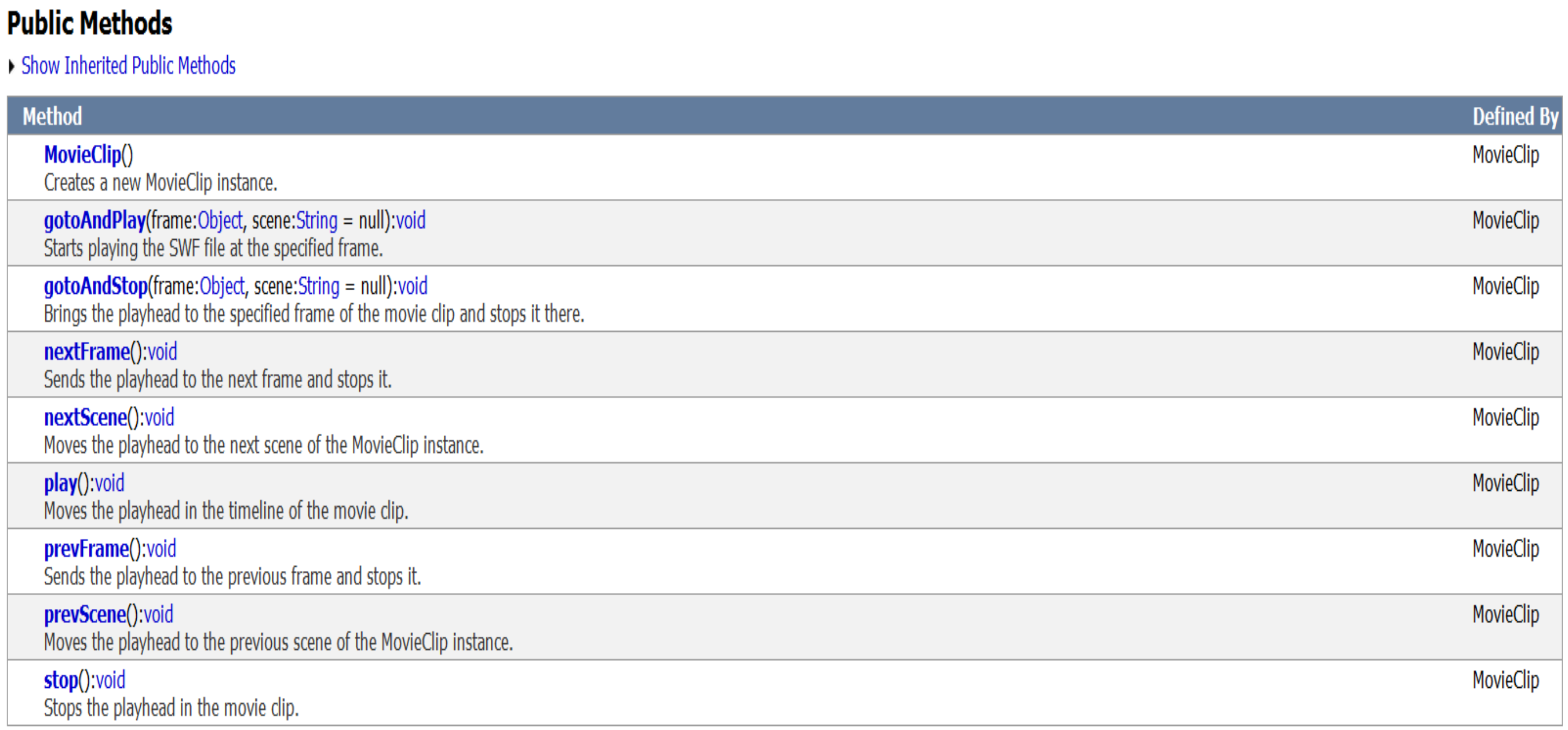
Public Properties

▸ Show Inherited Public Properties

YouTube - David Gilmour Marooned

Property	Defined By
currentFrame : int [read-only] Specifies the number of the frame in which the playhead is located in the timeline of the MovieClip instance.	MovieClip
currentFrameLabel : String [read-only] The label at the current frame in the timeline of the MovieClip instance.	MovieClip
currentLabel : String [read-only] The current label in which the playhead is located in the timeline of the MovieClip instance.	MovieClip
currentLabels : Array [read-only] Returns an array of FrameLabel objects from the current scene.	MovieClip
currentScene : Scene [read-only] The current scene in which the playhead is located in the timeline of the MovieClip instance.	MovieClip
enabled : Boolean A Boolean value that indicates whether a movie clip is enabled.	MovieClip
framesLoaded : int [read-only] The number of frames that are loaded from a streaming SWF file.	MovieClip
scenes : Array [read-only] An array of Scene objects, each listing the name, the number of frames, and the frame labels for a scene in the MovieClip instance.	MovieClip
totalFrames : int [read-only] The total number of frames in the MovieClip instance.	MovieClip
trackAsMenu : Boolean Indicates whether other display objects that are SimpleButton or MovieClip objects can receive mouse release events or other user input release events.	MovieClip

Attributes



Public Methods

▸ Show Inherited Public Methods

Method	Defined By
MovieClip() Creates a new MovieClip instance.	MovieClip
gotoAndPlay(frame:Object, scene:String = null):void Starts playing the SWF file at the specified frame.	MovieClip
gotoAndStop(frame:Object, scene:String = null):void Brings the playhead to the specified frame of the movie clip and stops it there.	MovieClip
nextFrame():void Sends the playhead to the next frame and stops it.	MovieClip
nextScene():void Moves the playhead to the next scene of the MovieClip instance.	MovieClip
play():void Moves the playhead in the timeline of the movie clip.	MovieClip
prevFrame():void Sends the playhead to the previous frame and stops it.	MovieClip
prevScene():void Moves the playhead to the previous scene of the MovieClip instance.	MovieClip
stop():void Stops the playhead in the movie clip.	MovieClip

Attributes

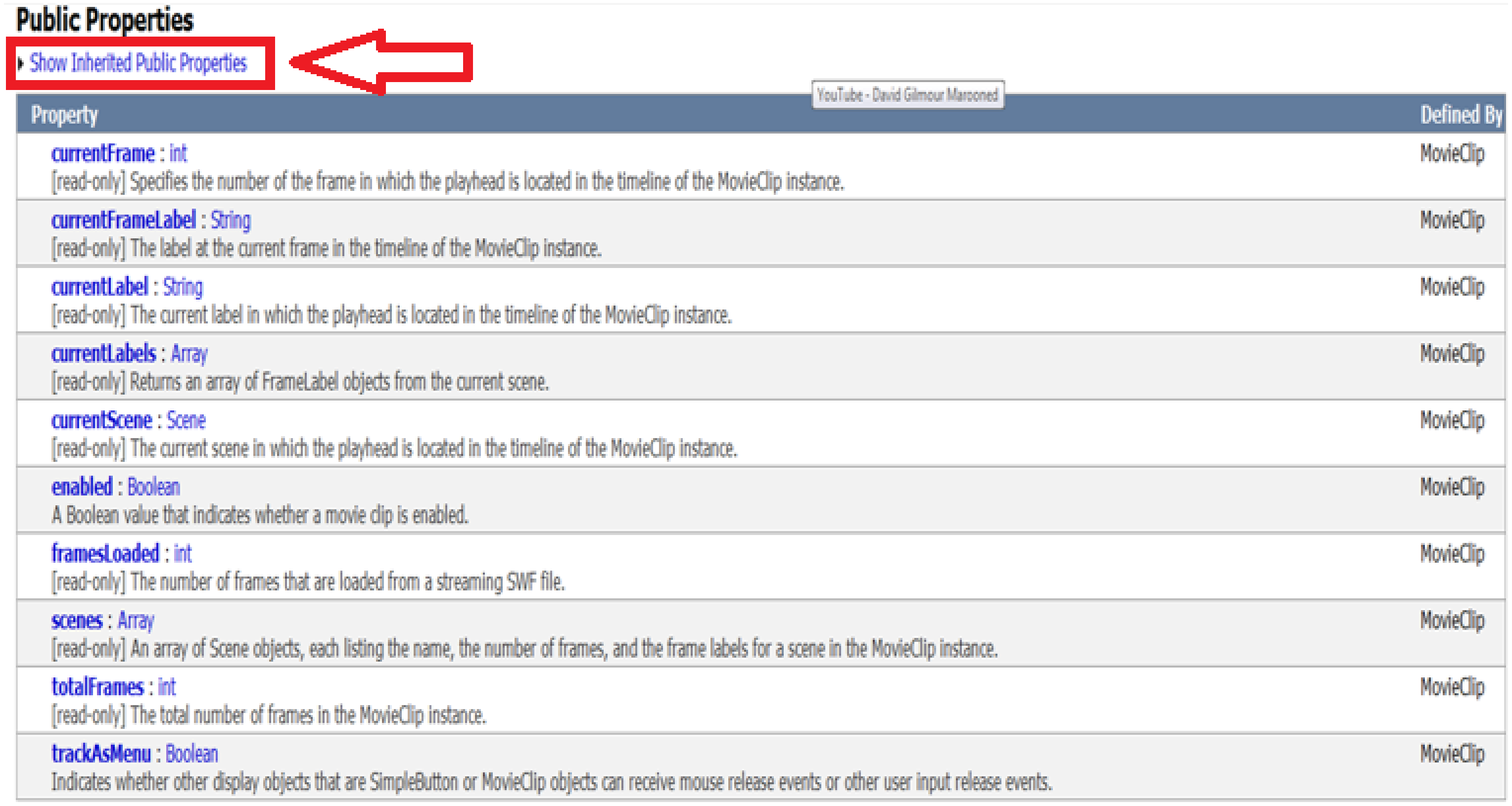
- Now as you can see, a lot of the attributes are not found in those two tables. Why is that?

- MovieClip inherits from other display objects, so the attributes that you saw in the previous two tables are the attributes unique for the MovieClip class.

- If you want to know all the attributes found inside the MovieClip class (which a lot of them where inherited from other display object classes), you should do one of the two things I will cover in the next two slides.

Attributes

- One way is to click on the “Show Inherited Public …” in the tables. It is found for properties and methods.



Public Properties

▸ Show Inherited Public Properties

YouTube - David Gilmour Marooned

Property	Defined By
currentFrame : int [read-only] Specifies the number of the frame in which the playhead is located in the timeline of the MovieClip instance.	MovieClip
currentFrameLabel : String [read-only] The label at the current frame in the timeline of the MovieClip instance.	MovieClip
currentLabel : String [read-only] The current label in which the playhead is located in the timeline of the MovieClip instance.	MovieClip
currentLabels : Array [read-only] Returns an array of FrameLabel objects from the current scene.	MovieClip
currentScene : Scene [read-only] The current scene in which the playhead is located in the timeline of the MovieClip instance.	MovieClip
enabled : Boolean A Boolean value that indicates whether a movie clip is enabled.	MovieClip
framesLoaded : int [read-only] The number of frames that are loaded from a streaming SWF file.	MovieClip
scenes : Array [read-only] An array of Scene objects, each listing the name, the number of frames, and the frame labels for a scene in the MovieClip instance.	MovieClip
totalFrames : int [read-only] The total number of frames in the MovieClip instance.	MovieClip
trackAsMenu : Boolean Indicates whether other display objects that are SimpleButton or MovieClip objects can receive mouse release events or other user input release events.	MovieClip

Attributes

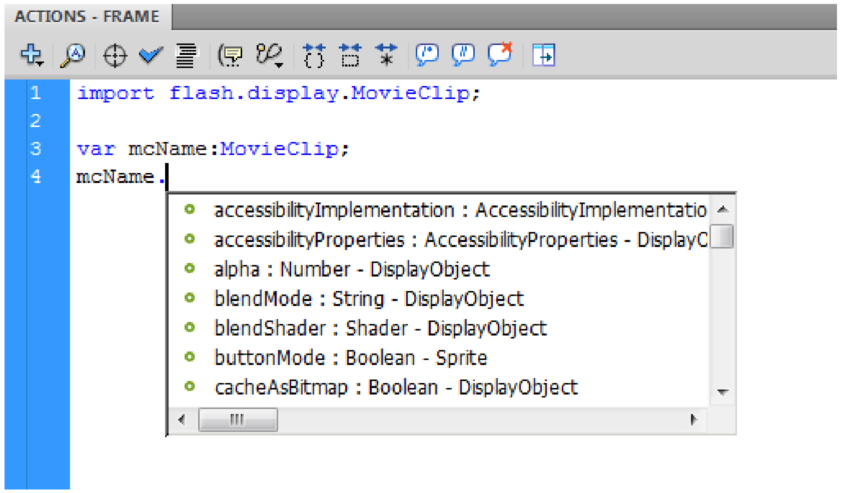
• If you don't want to go to the help, the 2nd solution to see all the attributes is done using the code.

- Anywhere in the code, create a "MovieClip" variable
  Example: ***var mcName:MovieClip;***

- Then, use the name followed by the dot "." operator.

- A list of attributes will be shown to you.

<u>**Note:**</u> When you create the variable of type movie clip, the following will be added "import flash.display.MovieClip;". It's ok, that is the file that contains all the MovieClip functionalities.



```
import flash.display.MovieClip;

var mcName:MovieClip;
mcName.
```

```
accessibilityImplementation : AccessibilityImplementatio
accessibilityProperties : AccessibilityProperties - DisplayC
alpha : Number - DisplayObject
blendMode : String - DisplayObject
blendShader : Shader - DisplayObject
buttonMode : Boolean - Sprite
cacheAsBitmap : Boolean - DisplayObject
```

Attributes

•For now, I talked about the x, y, scaleX, scaleY, rotation properties…

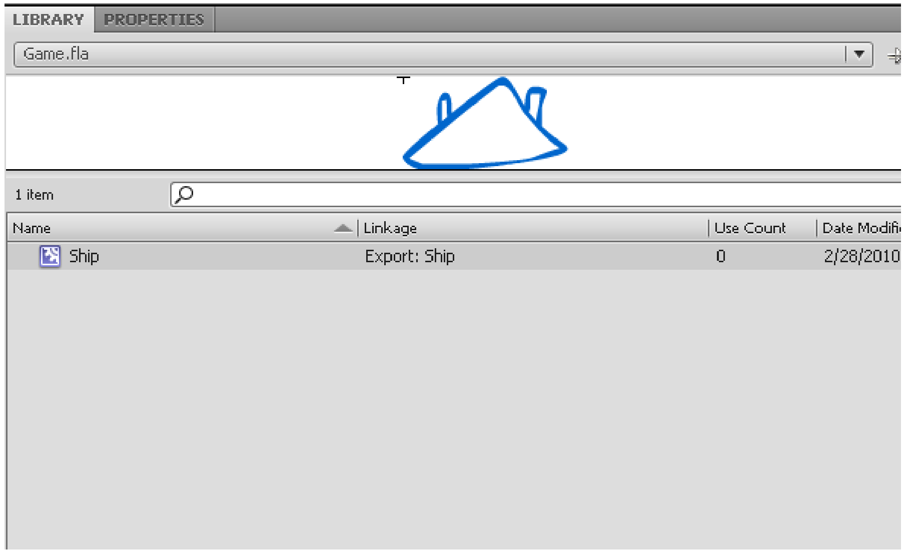
• I will leave it up to you to play around with more properties and attributes (at least the simple ones like: alpha, name, addChild, addChildAt …).

• In some of the coming chapters, I will be covering more attributes found in the MovieClip class (just because I have to ☺)

Adding Properties

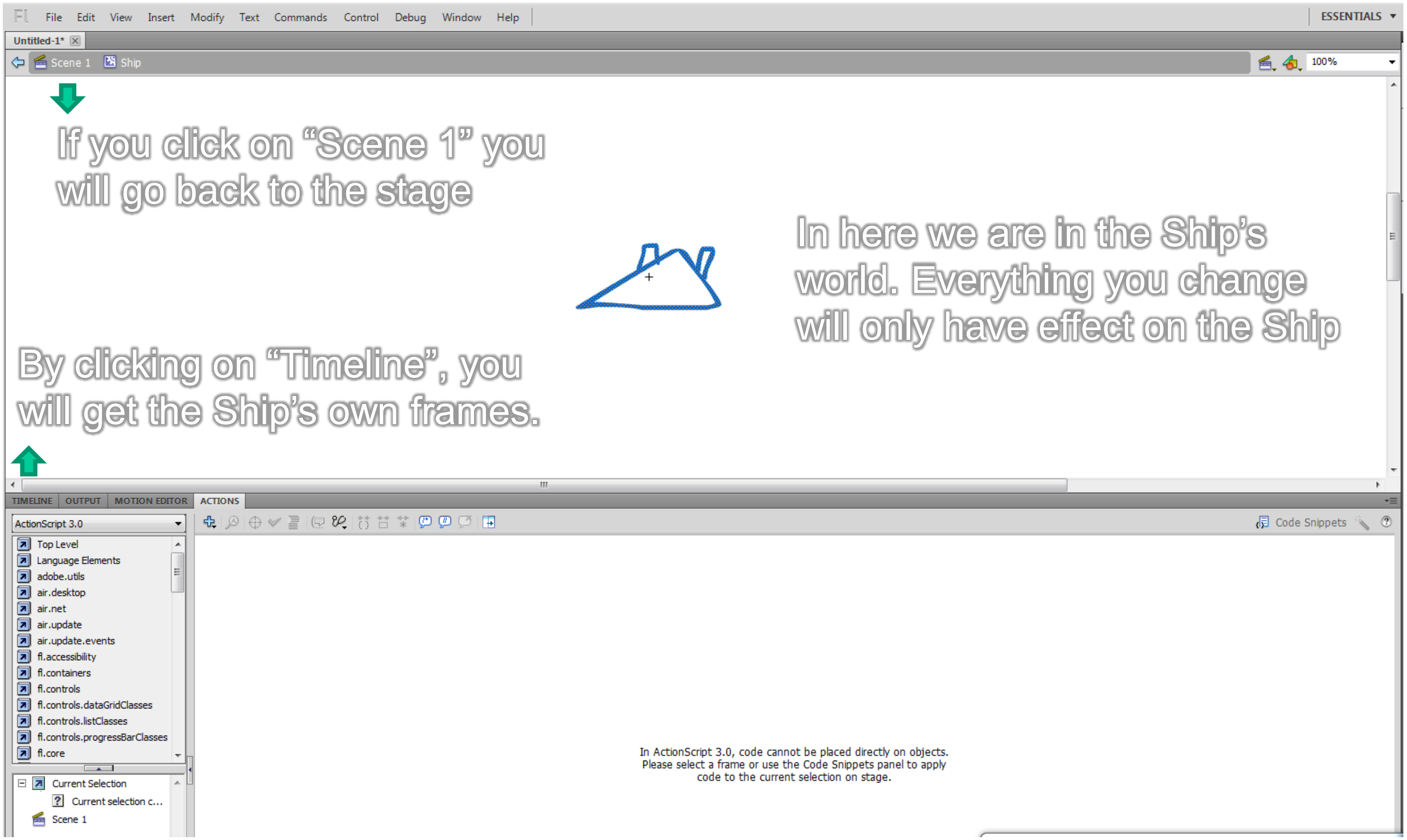
- Now, we know that our Ship class extends from MovieClip so it has access to all the attributes found in the MovieClip class. But how can we add extra attributes to our Ship class?? (health, Lives, Weapons, Move function …)

- Under Library, double click on the Ship class.



Name	Linkage	Use Count	Date Modifi
Ship	Export: Ship	0	2/28/2010

Adding Properties



If you click on "Scene 1" you will go back to the stage

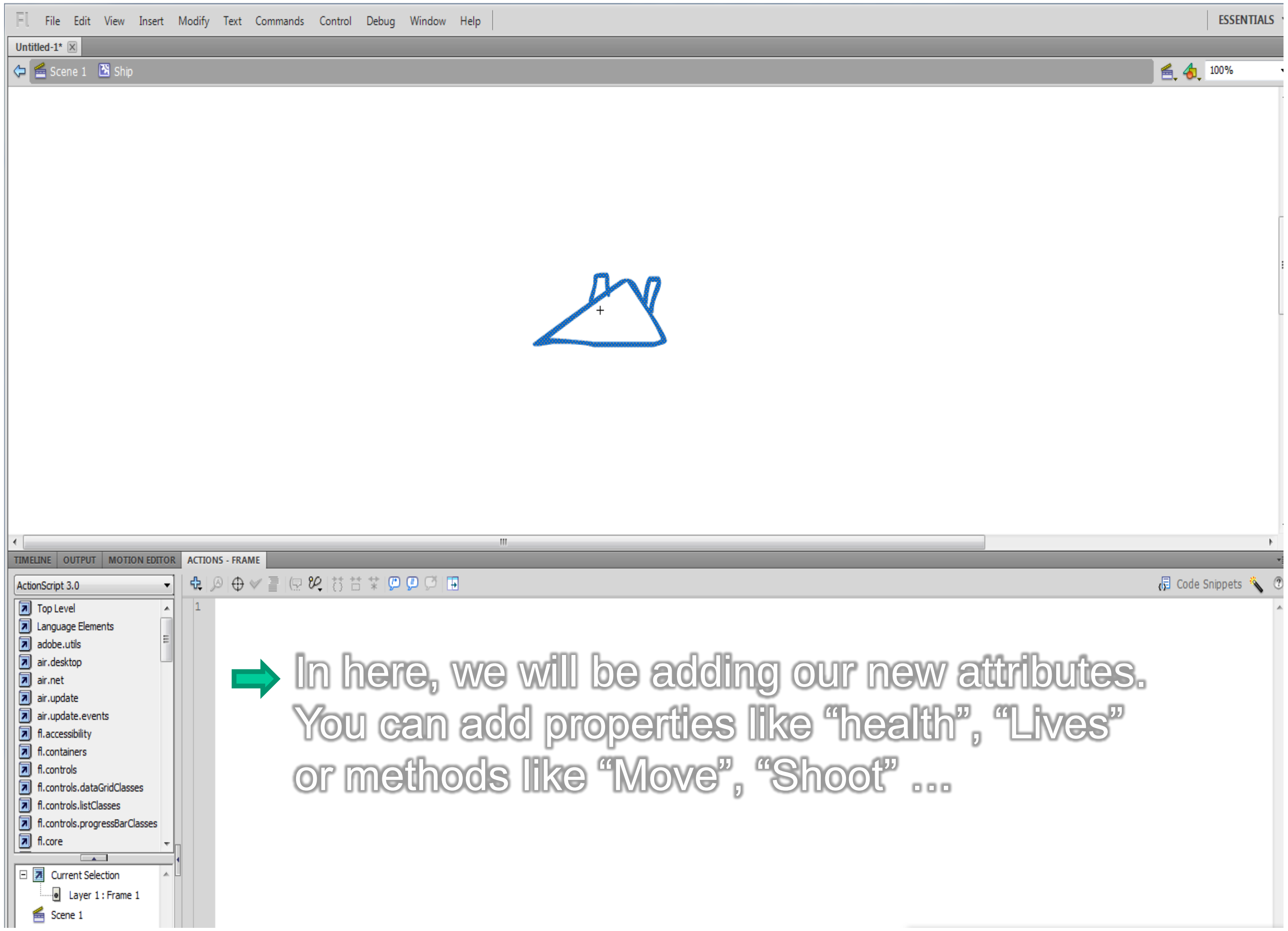
In here we are in the Ship's world. Everything you change will only have effect on the Ship

By clicking on "Timeline", you will get the Ship's own frames.

In ActionScript 3.0, code cannot be placed directly on objects.
Please select a frame or use the Code Snippets panel to apply
code to the current selection on stage.

Adding Properties

- Click on "Timeline".
- Click on the first frame.
- Then Click on "Actions"



In here, we will be adding our new attributes. You can add properties like "health", "Lives" or methods like "Move", "Shoot" …

Adding Properties

• Now, Let’s add a health variable for our ship:

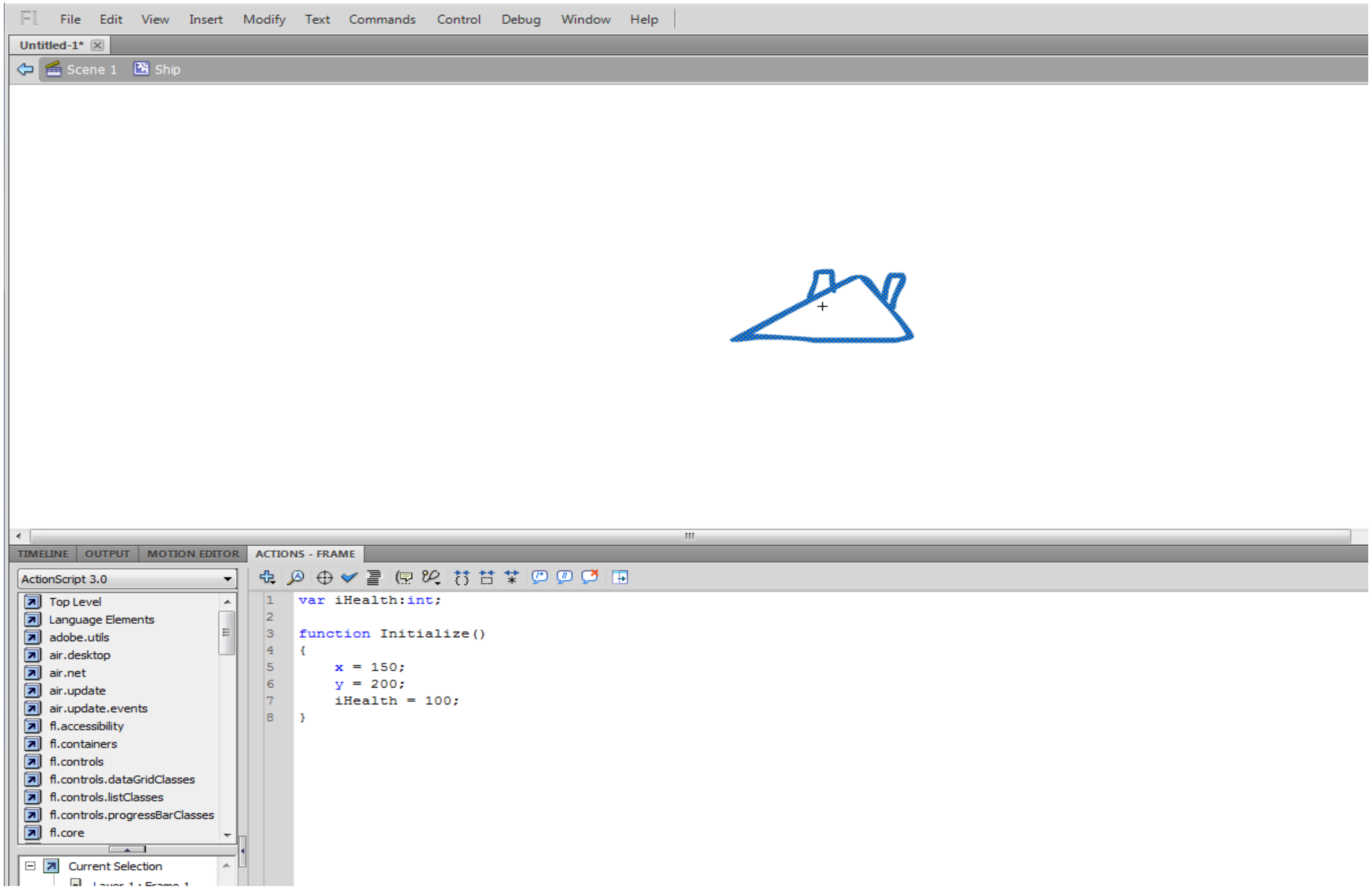
```
var iHealth:int;
```

<u>**Note:**</u> Do **NOT** initialize the variable to a value. You can set a value for it later, or we can create a special “Initialize” function for our character.

• Let’s create the “Initialize” function which will put our Ship at (150,200) and set the health to 100.

```
function Initialize()
{
        x = 150;
        y = 200;
        iHealth = 100;
}
```

Adding Properties



```
var iHealth:int;

function Initialize()
{
    x = 150;
    y = 200;
    iHealth = 100;
}
```

Adding Properties

- Click on "Scene 1" in the top left corner, so that we can dynamically create a ship and use the new properties.

- In the stage's actions tab, add the following code:

```
var mcShip:Ship = new Ship();
trace("x = " + mcShip.x);   /* x = 0 */
trace("y = " + mcShip.y);   /* y = 0 */
trace("health = " + mcShip.iHealth); /* health = 0 */

mcShip.Initialize();

trace("x = " + mcShip.x);  /* x = 150 */
trace("y = " + mcShip.y);  /* y = 200 */
trace("health = " + mcShip.iHealth);  /* health = 200 */
```

Note: Of course we can create more than one instance of the Ship class and each one of them will have it's own health value.

Adding Properties

- Finally, let's make our "Initialize" function more general.
- Let's say I want to create more than one ship but each one with a different location and health.
- We need to send parameters to our Initialize function.

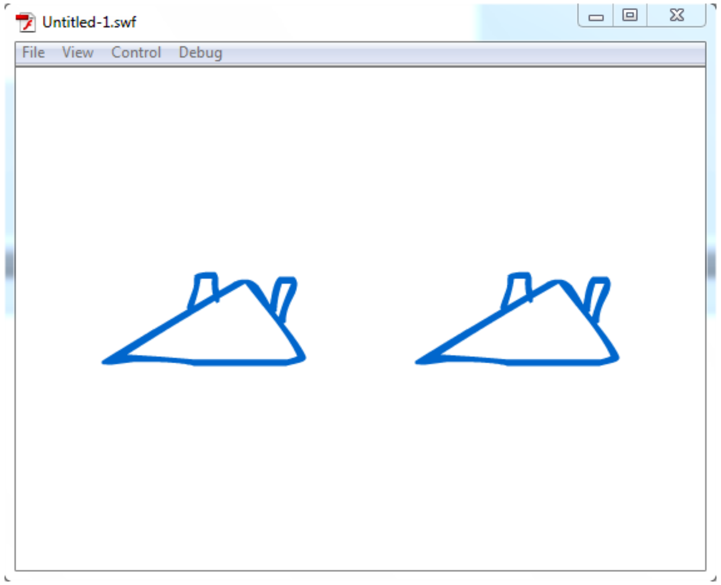
```
 function Initialize(iPosX:int , iPosY:int , iHealthValue:int)
{
   x = iPosX;
   y = iPosY;
   iHealth = iHealthValue;
}
```

Adding Properties

- Let's test our new "Initialize" function:

```
var mcShip1:Ship = new Ship();
mcShip1.Initialize(150,200,100);
stage.addChild(mcShip1);

var mcShip2:Ship = new Ship();
mcShip2.Initialize(400,200,50);
stage.addChild(mcShip2);
```



The End ☺