DigiPen INSTITUTE OF TECHNOLOGY

# CS 116 – Action Script Object Oriented Programming

Elie Abi Chahine

# Introduction

• Object-oriented programming (OOP) is a way of organizing the code in a program by grouping it into objects.

• Objects, are individual elements that include information (data values and functionality).

• Using an object-oriented approach to organizing a program allows you to group particular pieces of information (for example, music information like album title, track title, or artist name) together with common functionality or actions associated with that information (such as “add track to playlist” or “play all songs by this artist”). These items are combined into a single item, an object (for example, an “Album” or “MusicTrack”).

# Introduction

**Example in games:**

• So let's brainstorm and see how we can build an object for our character in the game.

• What are the character's properties?
- Shape
- Position
- Scale
- Rotation
- Speed
- Health
- ect…

• All those properties will be variables inside the object.

# Introduction

**Example in games:**

- Now, what are the character's functionalities or actions?
  - Move
  - Jump
  - Shoot
  - Play Animation
  - ect…
- All those functionalities(actions) will be methods(functions) inside the object.

# Attributes

- All the variables and methods inside an object are considered attributes

- For example:

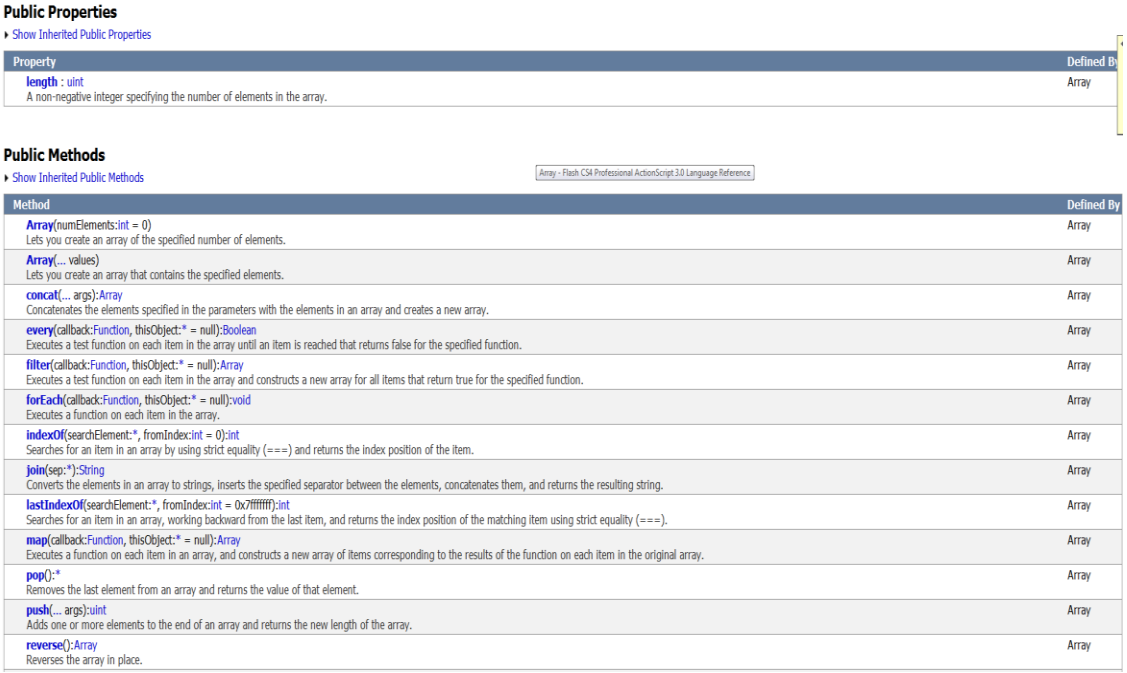
```
Health   →   var Health:int;

Jump     →   function Jump()
             {
                 .....
             }
```

- So when creating an object (a.k.a. class), you are creating a complex type that contains all those attributes.

# Attributes

- ActionScript presents to us a lot of objects, like Array, String, MovieClip …
- All those objects have attributes inside them.
- Example, for the Array, right click on the blue word “Array” then click on “View Help”. ***(You can do the same for all the types)***



**Public Properties**

Show Inherited Public Properties

| Property | Defined By |
|---|---|
| **length** : uint<br>A non-negative integer specifying the number of elements in the array. | Array |

**Public Methods**

Show Inherited Public Methods

Array - Flash CS4 Professional ActionScript 3.0 Language Reference

| Method | Defined By |
|---|---|
| **Array**(numElements:int = 0)<br>Lets you create an array of the specified number of elements. | Array |
| **Array**(... values)<br>Lets you create an array that contains the specified elements. | Array |
| **concat**(... args):Array<br>Concatenates the elements specified in the parameters with the elements in an array and creates a new array. | Array |
| **every**(callback:Function, thisObject:* = null):Boolean<br>Executes a test function on each item in the array until an item is reached that returns false for the specified function. | Array |
| **filter**(callback:Function, thisObject:* = null):Array<br>Executes a test function on each item in the array and constructs a new array for all items that return true for the specified function. | Array |
| **forEach**(callback:Function, thisObject:* = null):void<br>Executes a function on each item in the array. | Array |
| **indexOf**(searchElement:*, fromIndex:int = 0):int<br>Searches for an item in an array by using strict equality (===) and returns the index position of the item. | Array |
| **join**(sep:*):String<br>Converts the elements in an array to strings, inserts the specified separator between the elements, concatenates them, and returns the resulting string. | Array |
| **lastIndexOf**(searchElement:*, fromIndex:int = 0x7fffffff):int<br>Searches for an item in an array, working backward from the last item, and returns the index position of the matching item using strict equality (===). | Array |
| **map**(callback:Function, thisObject:* = null):Array<br>Executes a function on each item in an array, and constructs a new array of items corresponding to the results of the function on each item in the original array. | Array |
| **pop**():*<br>Removes the last element from an array and returns the value of that element. | Array |
| **push**(... args):uint<br>Adds one or more elements to the end of an array and returns the new length of the array. | Array |
| **reverse**():Array<br>Reverses the array in place. | Array |

# Object Instance

• So now, we know that we have objects in ActionScript. But how do we create a variable of that type?

Let's give an example with Arrays:

```
var aName:Array = new Array();
```

or

```
var aName:Array = new Array(10);
```

or

```
var aName:Array = new Array(1,2,3,4,5);
```

**Note:** I know we can do the following "***var aName:Array = [1,2,3,4];***" but that is a special case.

# Object Instance

**var aName:Array = new Array(…);**

**var** → Keyword in order to create a variable

**aName** → Variable's name

**Array** → Variable's Type

**new** → Operator that specifies we are creating an object

**Array(…)** → Constructor that will create our variable. (More on this in the next slide)

# Constructor

- The ***constructor*** would be the function that has the same name as the object type.

- It is called ***constructor*** because it is used to create (construct) the object.

- Some objects can have more than one constructor:

**Array**(numElements:int = 0)

- Lets you create an array of the specified number of elements.

**Array**(… values)

- Lets you create an array that contains the specified elements.

# Constructor

**For example:**

```
var aName:Array = new Array();
```

- Will create an empty Array with no elements inside

```
var aName:Array = new Array(10);
```

- Will create an array with 10 empty slots inside

```
var aName:Array = new Array(1,2,3,4,5);
```

- Will create an array with five slots filled with the values sent as parameter

**Note:** Check the constructors for each type in the help in order to know the different ways you can create an object of that type.

# Constructor

- As a last note, constructors are a special case of a function
  - They have to have the same name as the type
  - Don't have a return type. Well technically, their return type can only be the same type as the object, even though you don't specify it.
  - Can have parameters (as many as you want, with any type that you want, exactly like a normal function).
  - Called only at object creation.

# Dot Operator (.)

- The dot operator is used to access the attributes inside the object.
- For example:

```
var aMyArray:Array = new Array(1,2,3,4,5);
trace(aMyArray.length);    /* 5 */
trace(aMyArray);   /* 1,2,3,4,5 */

var aReMyArray:Array = aMyArray.reverse();
trace(aReMyArray); /* 5,4,3,2,1 */
```

# Examples Using Objects

**Example 1:** Array

```
var aNumbers:Array = new Array(1, 2, 3);
trace(aNumbers);       /* 1,2,3 */

var aLetters:Array = new Array("a", "b", "c");
trace(aLetters);       /* a,b,c */

var aNumbersAndLetters:Array = aNumbers.concat(aLetters);
trace(aNumbersAndLetters); /* 1,2,3,a,b,c */

var aLettersAndNumbers:Array = aLetters.concat(aNumbers);
trace(aLettersAndNumbers); /* a,b,c,1,2,3 */
```

# Examples Using Objects

**Example 2:** String

```
var sName:String = "JoHn";
var sLastName:String = "SmIth";

var sLowerName:String = sName.toLowerCase();
trace(sName);  /* JoHn */
trace(sLowerName);   /* john */

var sUpperLastName:String = sLastName.toUpperCase();
trace(sLastName);   /* SmIth */
trace(sUpperLastName);   /* SMITH */

var sFullName:String = sLowerName.toUpperCase() + sUpperLastName;
trace(sFullName);   /* JOHN SMITH */
```

# Exercises Using Objects

**Exercise 1:** Array

- Create an array containing 10 numbers (0 to 9)
- Using the methods inside the Array object, do the following:
  - Remove the last number from the list, then trace the result
  - Remove the first number from the list, then trace the result
  - Remove the numbers 5 and 6 from the list, then trace the result

# Exercises Using Objects

**Solution 1:** Array

```
var aMyArray:Array = new Array(0,1,2,3,4,5,6,7,8,9);
trace(aMyArray);

aMyArray.pop();
trace(aMyArray);

aMyArray.shift();
trace(aMyArray);

aMyArray.splice(4,2);
trace(aMyArray);
```

# Exercises Using Objects

**Exercise 2:** Date

- Using the Date object do the following:
  - Show the full Date and time (ex: Sat Oct 23 18:47:19 GMT-0700 2010 )
  - Trace the day (ex: Day: 6)
  - Trace the month (ex: Month: 9)
  - Trace the year (ex: 2010)

# Exercises Using Objects

**Solution 2:** Date

```
var aCurrentDate:Date = new Date();

trace(aCurrentDate);

trace(aCurrentDate.getDay());

trace(aCurrentDate.getMonth());

trace(aCurrentDate.getFullYear());
```

# The End ☺