More Flash and ActionScript

Khoo Yit Phang
CMSC 434
October 26, 2006
More Flash and ActionScript

1. Initializing in the First Frame
2. OOP in ActionScript
3. Combining Multiple Movies
Initializing in the First Frame

Setting up global data
Avoiding unintentional resets
Setting up global data

- Obvious solution: initialize in the very first frame
Unintentional data reset?

- What if you need to return to frame 1 (e.g. welcome screen)?
  - global data reinitialized!
Avoiding unintentional reset

- Solution:
  - leave initialization in the first frame
  - duplicate graphics/actions into the second frame
  - return to the second frame
Another way (very fragile!)

- Programmatically test for initialization (ala C/C++ #ifndef/#define/#endif):
  
  ```c
  if (_global.mydb == undefined) {
      /* do stuff */
      _global.mydb = ...;
  }
  ```

- Cannot declare functions inside an if-block!
  ... and a whole lot of other caveats.

- Simple data initialization is okay.
Object-Oriented Programming in ActionScript
OOP in ActionScript

- ActionScript has two different ways of creating objects:
  - old-styled *Prototype-based* objects
    (create an object and modify, identical to Javascript)
  - new-styled *Class-based* objects (similar to Java)

- Prototype-based objects are convenient for one-off instances (e.g. event handlers)
Prototype-based OO in AS

- Create a new Object() and modify:

```
var myObj = new Object();
myObj.myFunction = function() {
    this.myData = "data";
    /* Note: do not use myObj.myData in
    * in the body! Use this.myData instead */
}
```
Reusable Prototypes, part 1

1. Declare constructor function:
   ```javascript
   function MyObjProto(param1, param2) {
     /* initialize data members */
     this.data1 = param1;
     this.data2 = param2;
   }
   ```

2. Define prototype (member functions):
   ```javascript
   MyObjProto.prototype.myFunc = function() {
     /* member function for MyObjProto */
   }
   ```
3. Create an instance:

```javascript
var myObjInst: MyObjProto = new MyObjProto(param1, param2);
```

- Inheritance also possible by assigning to the `prototype` property of the constructor, or the `__proto__` property of an instance.

- **Reference (identical in Javascript):**

Combining Multiple Movies

Developing movies collaboratively
Why combine movies?

1. Perhaps you have multiple developers working on independent modules:
   - e.g. the sign-on and checkout pages of a shopping website

2. Or, you are reusing an existing movie:
   - e.g. a gallery of Flash games
Why not copy/paste?

- Turns out that you cannot copy/paste timelines
  - and copying individual keyframes is really painful and error-prone

- You may not have the source .fla file!
Example Scenario

- Say you have the above:
  - two Flash movies: MovieA, MovieB
  - one (shared) ActionScript module: SharedModule
- How to combine into a single application?
Combined Project

- Use a *container* movie (Container) to combine
  - Note: MovieA, MovieB **no longer include** the SharedModule; the container movie does instead
- Caveat: the two movies need to be compiled (.swf)
  - Tip: use the Project Panel in Flash to compile everything at once
All three movies have to be the same size

Container embeds MovieA, MovieB programmatically (after global data initialization):

```javascript
var loader = new MovieClipLoader();
var loaderevents = {};
loaderevents.onLoadInit = function(target:MovieClip) {
    target.show();
}
loader.loader.addListener(loaderevents);
loader.loader.loadClip("MovieA.swf", 1); // id 1
```
Implementation details 2

- MovieA, MovieB (and SharedModule) communicate through the _global namespace:

  Container:
  ```
  _global.msg = "";
  _global.switchToB = function() {
    /* _leveln refers to embedded movie with id n */
    _level1.hide();
    _level2.show();
  }
  ```
Implementation details 3

**MovieA:**

```javascript
function hide() { _visible = false; }
_global.msg = "some message";
_global.switchToB();
```

**MovieB:**

```javascript
function show() {
    _visible = true;
    doSomethingWith(_global.msg);
}
```