Memory management and optimization

Jun Heider, ACI+D
RealEyes Media
jun@realeyes.com

360|Flex Atlanta

About Me

Jun Heider

- Adobe Certified Flex developer and instructor
- I work for RealEyes Media in sunny Denver, CO http://www.realeyes.com
- I have worked on various Flex/AIR applications and consulting for Fortune 500 financial corporations, government, and many other private sector firms such as Chase Manhattan, US Navy, GLOBE/UCAR, and Beatport
- My Blog → http://www.iheartflex.com OR http://www.iheartair.com

About this presentation

- It is a 200 level presentation

- If you’re familiar with the works of Grant Skinner and Alex Harui, and use those techniques knowledgeably then this presso is NOT for you

- If you feel that you code at or above the level of guys like Doug Mccune, Darron Schall, and Michael Labriola, then this presso is NOT for you

- If you’re coming from a well versed programming background, much of this material will be similar to what you’ve seen in other OOP languages such as Java…however it will be shown in the light of Flex/AIR development

Housekeeping

- To ensure we cover all material in an efficient manner, we will attempt to save questions for the end.
- This material is not on your USB drives yet, however I will be delivering it to John and Tom later today. I will also post it to my blog.

Reasons for good memory management and optimization practices

- Speed
- Stability
- Standards

Outline

- Memory Management Topics
- Optimization Topics
- Tools
- Demos / Case Studies
- Q & A

MEMORY MANAGEMENT PRACTICES IN YOUR APPS

360|Flex Atlanta

Flash Player 9 was designed for performance, therefore the garbage collection system is not optimized and needs your help

How the garbage collector works

- **Runs only during allocation** – Flash Player grabs memory in chunks
- **Is unpredictable** – GC will run when the player determines it’s time, based on factors such as the number of object vs. available chunks and the amount of system RAM
- **Is automatic** – There is no supported way to force GC in a production app. Only one in debug mode and a Skinner hack
- **Is incremental** – When GC runs it may not do everything in one pass

How the garbage collector works

- When the garbage collector runs there’s two algorithms that it uses:
    - Incremental Mark and Sweep
    - Deferred Reference Counting

How the garbage collector works

- Incremental Mark and Sweep
    - Starts at garbage collection roots which can include the following:

      - Package-level variables
      - Local variables of a currently executing method or function
      - Static variables
      - Instance variables of the program's main class instance
      - Instance variables of an object on the Flash runtime display list
      *- Variables in the scope chain of a currently executing function or method**

        - Objects that can be reached via reference are "marked" to be kept
        - Objects that cannot be reached are "swept" since they are eligible for garbage collection

*http://www.moock.org/blog/archives/000223.html

How the garbage collector works

- Deferred Reference Counting
  - All *strong* references are counted against an object.
    - If the count is greater than 0 then the object is kept
    - If the count is 0 the object is eligible for garbage collection

How the garbage collector works

- Grant Skinner provided us with a very good demo on how the Flash Player garbage collection algorithms work:

http://www.gskinner.com/blog/archives/2006/09/garbage_collect.html

Best Practices

- We need to help the garbage collector by writing our code optimally

Best Practices

- If the object that contains the listener function will have other references to it you can safely use a weak reference:

```
// addEventListener(type:String, listener:Function,
//      useCapture:Boolean = false, priority:int = 0,
//      useWeakReference:Boolean = false):void
myObject.addEventListener("timer", timerHandler, false,0,true);
```

Best Practices

- If there is a chance that this will be the only reference to an object with a listener function, use a strong reference and make sure to removeEventListener() when appropriate:

```
// Strongly referenced listener
myObject.addEventListener("timer", timerHandler);

// Later on in the code
// removeEventListener(type:String, listener:Function,
//     useCapture:Boolean = false):void
myObject.removeEventListener("timer", timerHandler);
```

Best Practices

- Reduce Consumption
  - Defer instantiation until it is necessary
  - Reuse object instances rather than releasing and creating new ones

Best Practices

- When releasing an object
  - Remove any strong listeners pointing into it

```
myObject.removeEventListener("timer", timerHandler);
```

  - Clean up any variable references to it

```
myObject = null;
myArray = [];
```

Best Practices

- People get concerned about circular references, in the context of deferred reference counting.

```
myObject1 = {myDataGrid:new DataGrid()};

myObject2 = myObject1;

myObject1.object2 = myObject2;
```

- The good new is thanks to the other algorithm incremental mark and sweep, if both objects are no longer accessible via a garbage collection root, then the circular reference no longer has as much sting.

Good Memory Management is achievable by any developer by keeping the following in mind:

- Manage your references
- Manage your data allocation

OPTIMIZING YOUR APPS

Although the performance of Flash Player 9 is superb, there are things you can optimize to get every ounce of performance possible

Best Practices

- Don’t optimize too early
- Establish a baseline for comparison
- Make incremental changes
- Use source control!!!

Optimization Tips (Variables/Objects)

- Reuse as much as possible, instantiation is very costly!

```
var myStrongObj:MyObj = new MyObj("Hello",22.2);
```

- Strongly Type whenever possible

```
var myStrongObj:MyObj;
```

Optimization Tips (Lookup)

- If you’re going to look up something repeatedly, set a local var and reference that:
  - Class members

```
var member1:String = myStrongObj.member1;
```

  - Array Length

```
var aryLength:int = _myTestArray.length;
```

  - Static getters

```
var totalMem:uint = System.totalMemory;
```

Optimization Tip (Arrays/ArrayCollection)

- If you want to invoke a method on an array member, invoking it by casting it first is faster than accessing it through array notation.

```
var currentObj:MyObj = myArray[0];
currentObj.addNumbers(10,10);
```

- Keep your Arrays Dense.

```
myArray.push("jun heider");
```

Optimization Tips (Numbers and Math)

- Use int/uint vs. Number

```
var numericalValue1:int = 1;
```

- If it makes sense, use bitwise operators

```
var anAge:int = 32;

// results in 256 (Bits: 0000 1000 = 8)
mutiplyResult = anAge << 3;

// results in 1 (Bits: 0010 0000 = 32)
divideResult = anAge >> 5;
```

Optimization Tips (Error Handling)

- If a Try/Catch FAILS it takes much longer that standard conditional statements

```
try
{
  p_evt.result.getItemAt(0).getItemAt(0);
}
catch(e:Error)
{
  //...
}
```

```
if( p_evt.result != null
    && p_evt.result is Array
    && p_evt.result.getItemAt(0) is Array)
{
  //...
}
```

Optimization Tips (Graphics)

- Beware of over-zealous effects and animations, try turning down their speeds and other thing that can affect performance
- Use effectStart and effectEnd events to hide and redisplay parts of the view to speed up effects
- When moving display objects, set the object’s includeInLayout property to false until its in its final location

Optimization Tips (Flex Components)

- Use deferred instantiation:
  - Use TileList vs. Tile/Repeater
  - In Containers look for the creationPolicy attribute and set it to auto

Optimization Tips (Flex Components)

- Avoid unnecessary component nesting:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="vertical">
          <mx:VBox>
            <mx:HBox>
              <mx:ControlBar>
                <mx:Button label="button1"/>
                <mx:Button label="button2"/>
                <mx:Button label="button3"/>
              </mx:ControlBar>
            </mx:HBox>
          ...
          </mx:VBox>
</mx:Application>
```

Optimization Tips (Retrieving Remote Data)

- If you can use it AMF/Remoting is the way to go! (BlazeDS, LCDS, ColdFusion, AMFPHP, WebORB)
- James Ward has a nice comparison tool to do research on remote data retrieval protocols: http://www.jamesward.org/census/

Although Flash Player 9 is lighting fast, there are many ways that you can gain additional performance keeping in mind that other than basic best practices like removing unnecessary containers you should make changes incrementally and test results often.

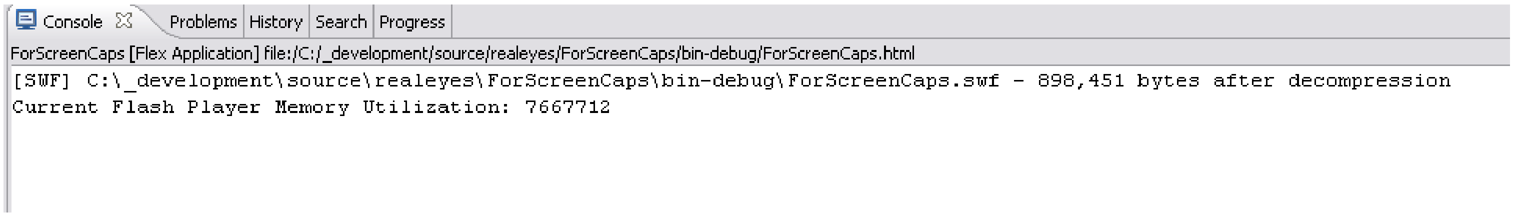
TOOLS

Tools

- Flash Player
  - For memory management:
    - System.totalMemory()
    - System.gc() (As of Flex 3, only in debug mode)
  - For performance tuning:
    - flash.utils.getTimer():int

Tools (Flex Builder)

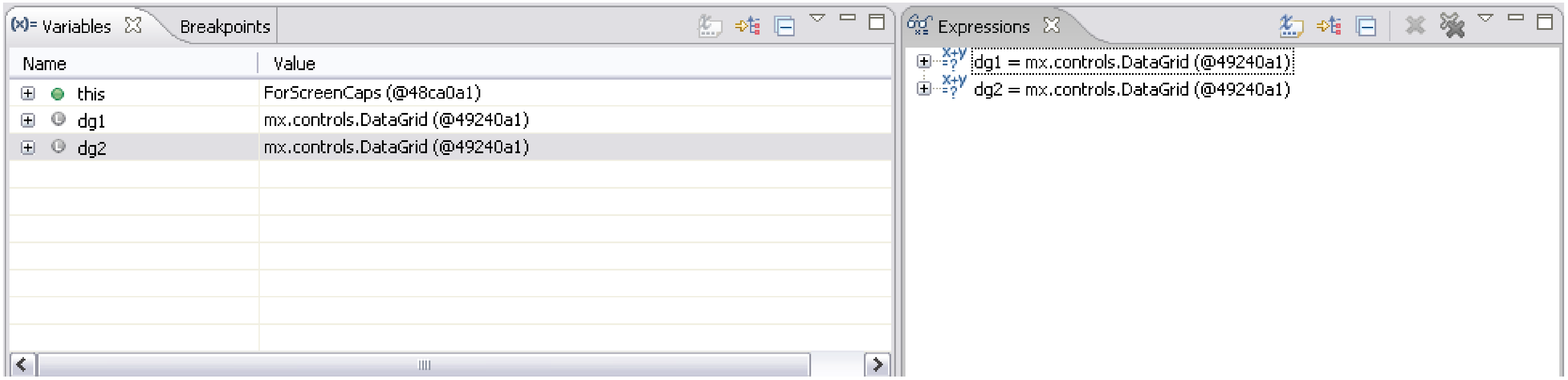
- Console Output View



```
ForScreenCaps [Flex Application] file:/C:/_development/source/realeyes/ForScreenCaps/bin-debug/ForScreenCaps.html
[SWF] C:\_development\source\realeyes\ForScreenCaps\bin-debug\ForScreenCaps.swf - 898,451 bytes after decompression
Current Flash Player Memory Utilization: 7667712
```

Tools (Flex Builder)

- Debug Perspective

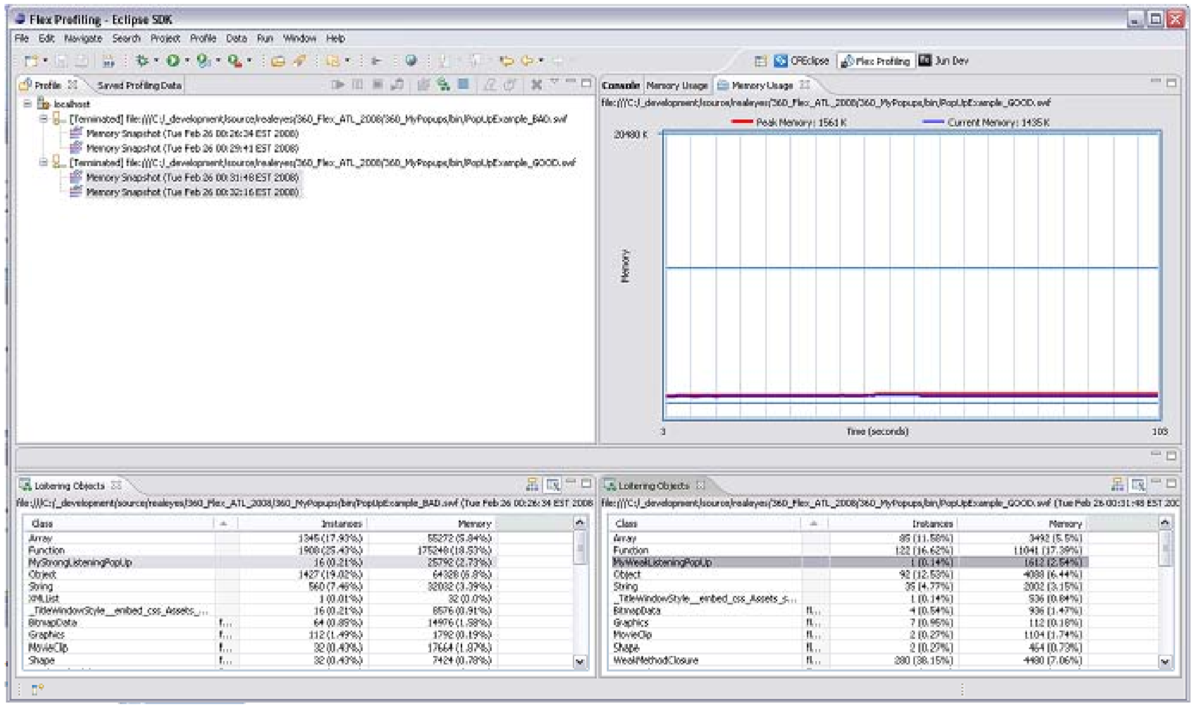


Name	Value
this	ForScreenCaps (@48ca0a1)
dg1	mx.controls.DataGrid (@49240a1)
dg2	mx.controls.DataGrid (@49240a1)

```
dg1 = mx.controls.DataGrid (@49240a1)
dg2 = mx.controls.DataGrid (@49240a1)
```

Tools (Flex Builder)

- Performance Profiler (As of Flex 3)



Flex Profiling - Eclipse SDK
File Edit Navigate Search Project Profile Data Run Window Help
CFEclipse Flex Profiling Jun Dev
Profile Saved Profiling Data
Console Memory Usage Memory Usage
Peak Memory: 1561 K
Current Memory: 1435 K
20480 K
Memory
3
Time (seconds)
103
Loitering Objects

localhost
[Terminated] file:///C:/_development/source/realeyes/360_Flex_ATL_2008/360_MyPopups/bin/PopUpExample_BAD.swf
Memory Snapshot (Tue Feb 26 00:26:34 EST 2008)
Memory Snapshot (Tue Feb 26 00:29:41 EST 2008)
[Terminated] file:///C:/_development/source/realeyes/360_Flex_ATL_2008/360_MyPopups/bin/PopUpExample_GOOD.swf
Memory Snapshot (Tue Feb 26 00:31:48 EST 2008)
Memory Snapshot (Tue Feb 26 00:32:16 EST 2008)

file:///C:/_development/source/realeyes/360_Flex_ATL_2008/360_MyPopups/bin/PopUpExample_GOOD.swf

file:///C:/_development/source/realeyes/360_Flex_ATL_2008/360_MyPopups/bin/PopUpExample_BAD.swf (Tue Feb 26 00:26:34 EST 2008

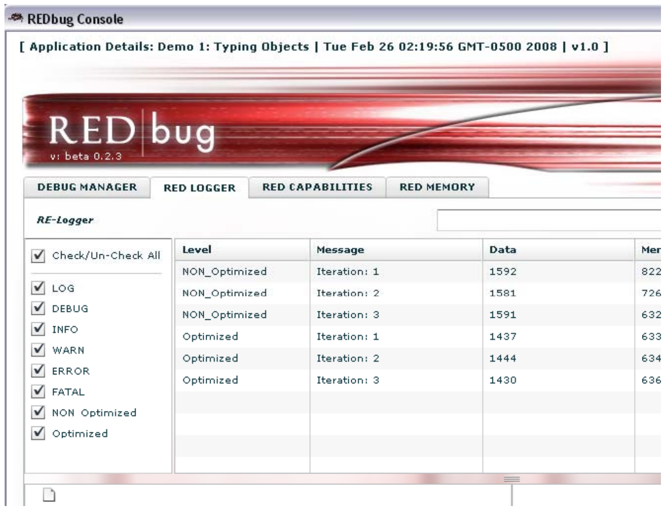
Class		Instances	Memory
Array		1346 (17.93%)	55272 (5.84%)
Function		1908 (25.43%)	175248 (18.53%)
MyStrongListeningPopUp		16 (0.21%)	25792 (2.73%)
Object		1427 (19.02%)	64328 (6.8%)
String		560 (7.46%)	32032 (3.39%)
XMLList		1 (0.01%)	32 (0.0%)
_TitleWindowStyle__embed_css_Assets_...		16 (0.21%)	8576 (0.91%)
BitmapData	f...	64 (0.85%)	14976 (1.58%)
Graphics	f...	112 (1.49%)	1792 (0.19%)
MovieClip	f...	32 (0.43%)	17664 (1.87%)
Shape	f...	32 (0.43%)	7424 (0.78%)

file:///C:/_development/source/realeyes/360_Flex_ATL_2008/360_MyPopups/bin/PopUpExample_GOOD.swf (Tue Feb 26 00:31:48 EST 200

Class		Instances	Memory
Array		85 (11.58%)	3492 (5.5%)
Function		122 (16.62%)	11041 (17.39%)
MyWeakListeningPopUp		1 (0.14%)	1612 (2.54%)
Object		92 (12.53%)	4088 (6.44%)
String		35 (4.77%)	2002 (3.15%)
_TitleWindowStyle__embed_css_Assets_s...		1 (0.14%)	536 (0.84%)
BitmapData	fl...	4 (0.54%)	936 (1.47%)
Graphics	fl...	7 (0.95%)	112 (0.18%)
MovieClip	fl...	2 (0.27%)	1104 (1.74%)
Shape	fl...	2 (0.27%)	464 (0.73%)
WeakMethodClosure	fl...	280 (38.15%)	4480 (7.06%)

Tools (REDbug, in extended beta…)

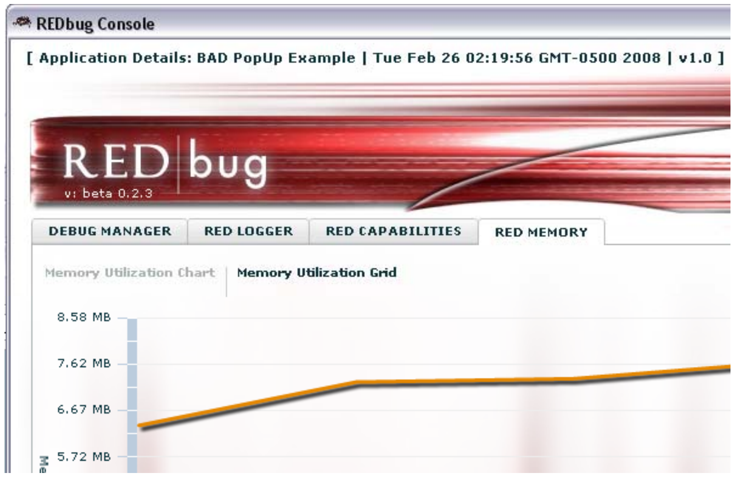
- Using some REDbug libraries, allows you to trace and get memory stats without having to run in debug mode


REDbug Console

[Application Details: Demo 1: Typing Objects | Tue Feb 26 02:19:56 GMT-0500 2008 | v1.0]

✓ Check/Un-Check All
✓ LOG
✓ DEBUG
✓ INFO
✓ WARN
✓ ERROR
✓ FATAL
✓ NON Optimized
✓ Optimized

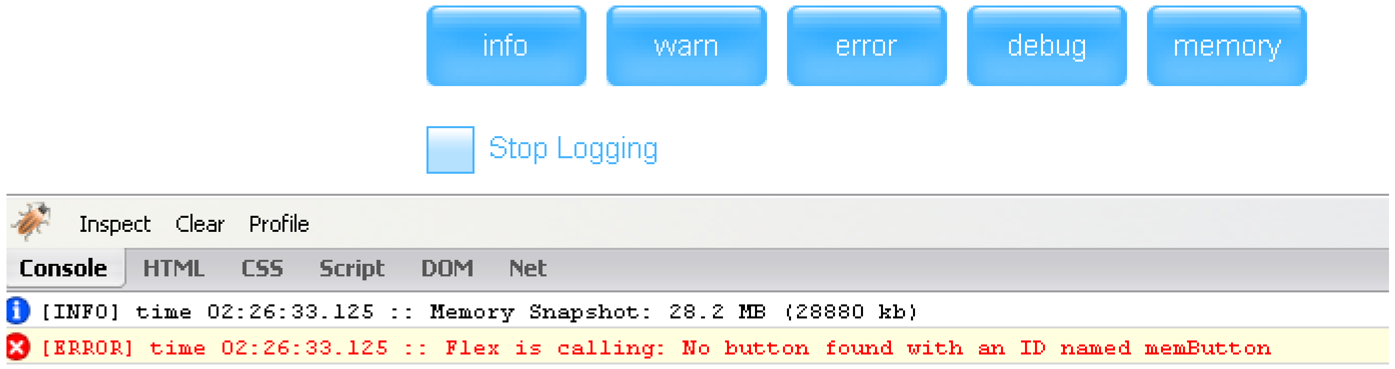
Level	Message	Data	Mer
NON_Optimized	Iteration: 1	1592	822
NON_Optimized	Iteration: 2	1581	726
NON_Optimized	Iteration: 3	1591	632
Optimized	Iteration: 1	1437	633
Optimized	Iteration: 2	1444	634
Optimized	Iteration: 3	1430	636


REDbug Console

[Application Details: BAD PopUp Example | Tue Feb 26 02:19:56 GMT-0500 2008 | v1.0]

Tools (Firebug/Thunderbolt)

- Works inside the Firebug plugin for Firefox



```
[INFO] time 02:26:33.125 :: Memory Snapshot: 28.2 MB (28880 kb)
[ERROR] time 02:26:33.125 :: Flex is calling: No button found with an ID named memButton
```

Tools

- Adobe Bugbase:

  http://bugs.adobe.com/flex

- Flexcoders:

  http://tech.groups.yahoo.com/group/flexcoders/

JUN PUTS STUFF TO THE TEST (DEMOS/CASE STUDIES)

IN CLOSING

Physical References:

- Essential ActionScript 3.0 (Moock)
  http://www.oreilly.com/catalog/9780596526948/

- Garbage Collection: Algorithms for Automatic Dynamic Memory Management (Not for the faint of heart)
  http://www.amazon.com/Garbage-Collection-Algorithms-Automatic-Management/dp/0471941484/ref=pd_bbs_sr_1?ie=UTF8&s=books&qid=1204011481&sr=8-1

Online References:

Tools:

- REDbug (Currently in need of official update to AIR 1.0, later this week, in the meantime use the one provided with the 360Flex USB resources): http://www.redbugtool.com/
- Thunderbolt: http://code.google.com/p/flash-thunderbolt/
- Firebug: https://addons.mozilla.org/en-US/firefox/addon/1843

Online References:

Memory Management:

- Grant Skinner: http://www.gskinner.com/blog/archives/2006/06/as3_resource_ma.html

- Alex Harui: http://blogs.adobe.com/aharui/2007/03/garbage_collection_and_memory.html

- Kyle Quevillon: http://blog.flexmonkeypatches.com/2007/03/28/flash-player-memory-management-and-garbage-collection-redux-2/

Online References:

Others:

- Tamarin (AVM2) MMgc: http://developer.mozilla.org/en/docs/MMgc
- AVM2 Overview: http://www.adobe.com/devnet/actionscript/articles/avm2overview.pdf
- Old Devnet Article on Flex performance: http://www.adobe.com/devnet/flex/articles/client_perf.html
- My blog (I'll have some more resource links I'll post up this week): http://www.iheartair.com OR http://www.iheartflex.com

Any Questions?