Advanced Desktop Development with Adobe AIR

Mike Chambers

Principal Product Manager
Flash Platform

Overview

- File Monitoring

- Volume Monitoring

- Data / Object Persistence

- Caching Assets for offline and performance

- Syncing offline data online

- Working with native applications

- AIR 2.0 Sneaks

as3corelib

as3corelib – Project Hosting on Google Code

http://code.google.com/p/as3corelib/

AIR 2.0   Flex 3.4   Delicious

as3corelib
ActionScript 3.0 library for several basic utilities.

Project Home   **Downloads**   **Wiki**   **Issues**   **Source**
Summary | Updates | People

The corelib project is an ActionScript 3 Library that contains a number of classes and utilities for working with ActionScript 3. These include classes for MD5 and SHA 1 hashing, Image encoders, and JSON serialization as well as general String, Number and Date APIs.

Search projects

Code license: New BSD License

Labels: actionscript, as3, library, api, adobe, corelib, md5, json, parsing, png, jpg, encryption

Featured downloads: Show all
⬇ as3corelib-.92.1.zip

Featured wiki pages: Show all
Main
ReleaseNotes
Resources
RoadMap

Links: ActionScript 3.0 Libraries Project Home

Feeds: Project feeds

Groups: Core Library Group

File Monitoring

- Detect when files are created, deleted and / or modified

- No built in way to monitor files

- Can be difficult due to possibility of file being deleted, moved, etc...

as3corelib : FileMonitor

```actionscript
import flash.filesystem.File;

private var monitor:FileMonitor;
private function onClick(e:Event):void
{
    var f:File = File.desktopDirectory.resolvePath("file.text");

    monitor = new FileMonitor();
    monitor.addEventListener(FileMonitorEvent.CHANGE, onFileChange);
    monitor.addEventListener(FileMonitorEvent.MOVE, onFileMove);
    monitor.file = f;
    monitor.watch();
}

private function onFileMove(e:FileMonitorEvent):void
{
    trace("file moved");
}

private function onFileChange(e:FileMonitorEvent):void
{
    trace("file changed");
}
```

Volume Monitoring

Detect when a volume is added or removed from system

- USB and Firewire Drives

- Cameras / Mobile Devices (that mount as drives)

- CD / DVD Roms

Volume Monitoring

File.getRootDirectories() is inconsistent across platforms

Windows :
   Array of Root Drives / Files

Mac / Linux :
   Array with single root "/" File

as3corelib : VolumeMonitor

```actionscript
private var monitor:VolumeMonitor;
private function onStartMonitoring():void
{
    monitor = new VolumeMonitor();
    monitor.addEventListener(FileMonitorEvent.ADD_VOLUME, onAddVolume);
    monitor.addEventListener(FileMonitorEvent.REMOVE_VOLUME, onRemoveVolume);

    monitor.watch();
}

private function onAddVolume(e:FileMonitorEvent):void
{
    trace("Volume added");
}

private function onRemoveVolume(e:FileMonitorEvent):void
{
    trace("Volume removed");
}
```

"Athena"

"Athena" : StorageVolume APIs

- "Athena" API

- Monitors mounting and un-mounting of storage volumes

- USB / Firewire

- Cameras and Devices that mount as drives

"Athena" : StorageVolume Events

```
var storageVolumeInfo = StorageVolumeInfo.storageVolumeInfo;

storageVolumeInfo.addEventListener(
                    StorageVolumeChangeEvent.STORAGE_VOLUME_MOUNT,
                    onVolumeMount);

storageVolumeInfo.addEventListener(
                    StorageVolumeChangeEvent.STORAGE_VOLUME_UNMOUNT,
                    onVolumeUnmount);
```

Caching Assets

- Enable for offline use

- Improved performance even when online

- Useful for images, data which does not degrade with age

as3corelib : ResourceCache

```actionscript
import com.adobe.air.net.ResourceCache;
import com.adobe.air.net.events.ResourceCacheEvent;

private static const CACHE_NAME:String = "cacheexample";
private var cache:ResourceCache;

private function onLoadImageClick():void
{
    if(!cache)
    {
        cache = new ResourceCache(CACHE_NAME);
        cache.addEventListener(ResourceCacheEvent.ITEM_READY, onItemReady);
    }

    cache.retrieve(urlField.text);

}

private function onItemReady(e:ResourceCacheEvent):void
{
    image.source = e.file.url;
}
```
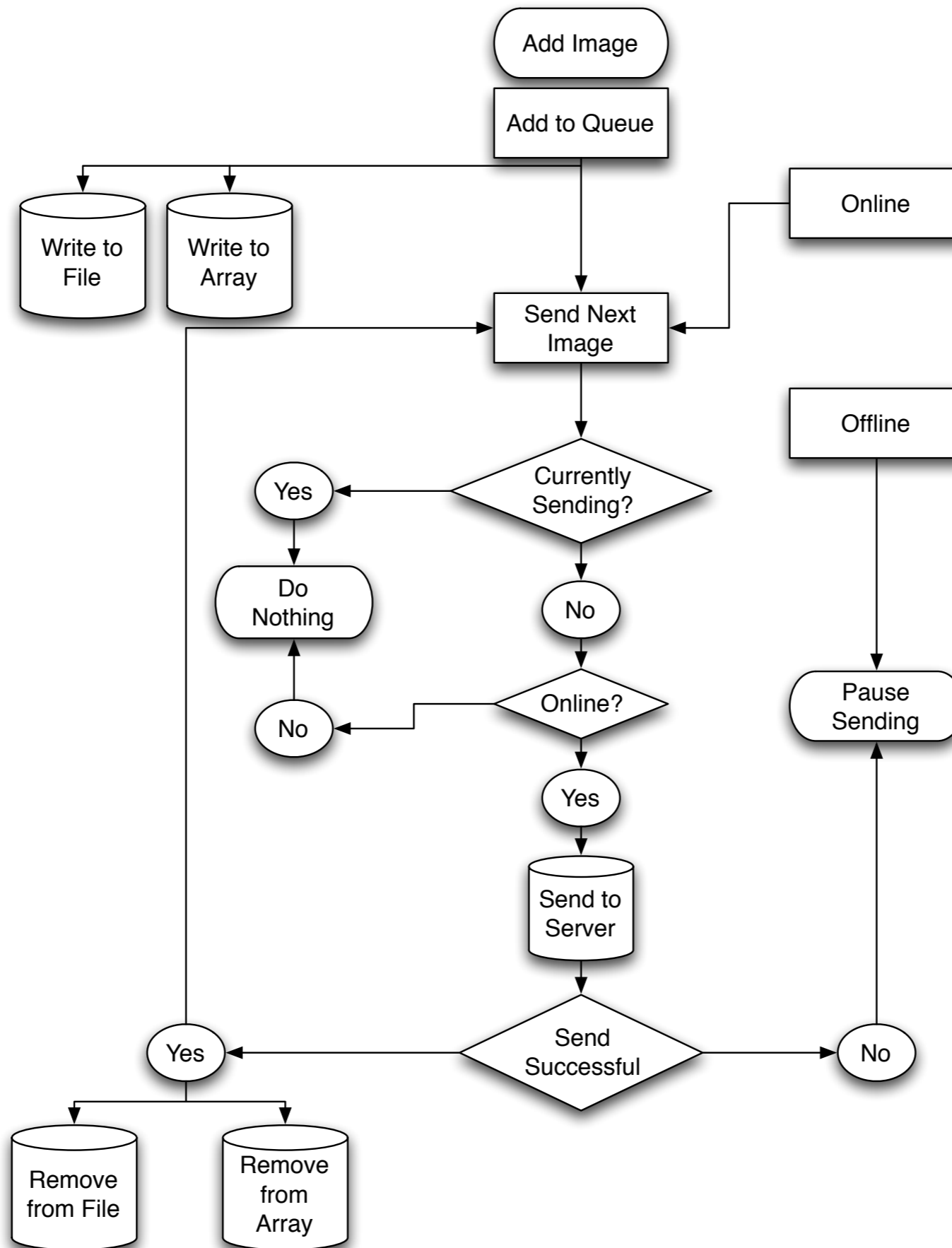
Data / Object Persistence

- Archive Objects directly to File system

- Easy to implement

- Can be used as custom file format for application

Syncing Offline Data

- Sync local data additions / changes to server

- Useful for offline apps, or applications that may have poor connections

```
                                    ┌──────────────┐
                                    │  Add Image   │
                                    └──────┬───────┘
                                           │
                                    ┌──────┴───────┐
                                    │ Add to Queue │
                                    └──────┬───────┘
        ┌──────────────┬──────────────────┤                    ┌──────────────┐
        │              │                  │                    │    Online    │
  ┌─────┴─────┐  ┌─────┴─────┐            │                    └──────┬───────┘
  │ Write to  │  │ Write to  │     ┌──────┴───────┐                   │
  │   File    │  │   Array   │     │  Send Next   ├───────────────────┘
  └───────────┘  └─────┬─────┘     │    Image     │
                       │           └──────┬───────┘
                       │                  │
                       │         ┌────────┴────────┐          ┌──────────────┐
         ┌───────┐     │         │    Currently    │          │   Offline    │
         │  Yes  │◄──────────────┤    Sending?     │          └──────┬───────┘
         └───┬───┘     │         └────────┬────────┘                 │
             │         │                  │                          │
       ┌─────┴─────┐   │              ┌───┴───┐                ┌──────┴───────┐
       │    Do     │   │              │  No   │                │    Pause     │
       │  Nothing  │   │              └───┬───┘                │   Sending    │
       └─────┬─────┘   │                  │                    └──────┬───────┘
             │         │         ┌────────┴────────┐                  │
         ┌───┴───┐     │         │     Online?     │◄─────────┐       │
         │  No   │◄──────────────┤                 │          │       │
         └───────┘     │         └────────┬────────┘          │       │
                       │              ┌───┴───┐               │       │
                       │              │  Yes  │               │       │
                       │              └───┬───┘               │       │
                       │           ┌──────┴───────┐           │       │
                       │           │   Send to    │           │       │
                       │           │    Server    │           │       │
                       │           └──────┬───────┘           │       │
                       │         ┌────────┴────────┐          │       │
         ┌───────┐     │         │      Send       │     ┌────┴───┐   │
         │  Yes  │◄──────────────┤   Successful    ├────►│   No   ├───┘
         └───┬───┘     │         └─────────────────┘     └────────┘
      ┌──────┴──────┐
 ┌────┴────┐  ┌─────┴─────┐
 │ Remove  │  │  Remove   │
 │from File│  │   from    │
 │         │  │   Array   │
 └─────────┘  └───────────┘
```

Working with Native Processes / Applications

- Currently no formal API for working with native processes

- CommandProxy Pattern : extend via proxy / helper app

- Requires custom installer
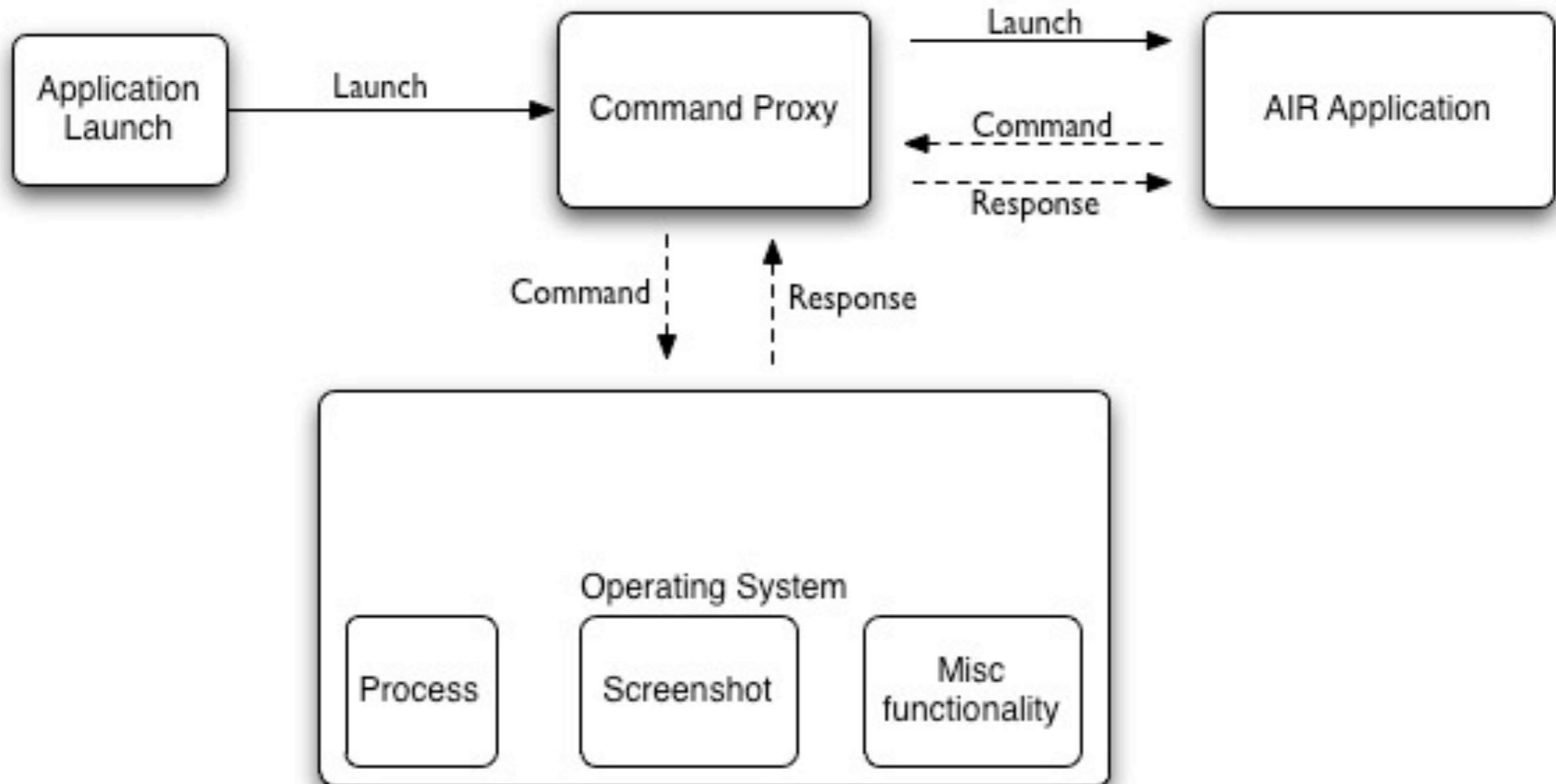
- Can use Adobe AIR silent install

CommandProxy

- Proof of concept

- Written in C#, ActionScript 3.0

- Could be written in any desktop language

Command Proxy Lifecycle

Application Launch → **Launch** → Command Proxy → **Launch** → AIR Application

Command Proxy ← **Command** — AIR Application

Command Proxy — **Response** → AIR Application

Command Proxy → **Command** → Operating System

Operating System → **Response** → Command Proxy

Operating System

- Process
- Screenshot
- Misc functionality

"Athena" : NativeProcess API

- New API in AIR 2.0

- Can call and communicate with external applications

- Requires application be distributed as native installer (no AIR files)

- Cannot execute applications within application directory

- Must add "extendedDesktop" to support profiles

"Athena" : NativeProcess

```
var app:File = new File(APP_EXECUTABLE_PATH);

var process:NativeProcess = new NativeProcess();

var args:Vector.<String>;args = new Vector.<String>(2, true);

args[0] = "-a";
args[1] = "foo";

var startupInfo:NativeProcessStartupInfo = new NativeProcessStartupInfo();

startupInfo.arguments = args;
startupInfo.executable = app;

process.start(startupInfo);
```

AIR 2.0 : Launch Default

```
var fileToOpen:File = File.desktopDirectory.resolvePath("readme.txt");
fileToOpen.openWithDefaultApplication();
```

- Allows file to be launched with default application

- Works in any AIR application

- File cannot be in application directory

- File type blacklist

Mike Chambers
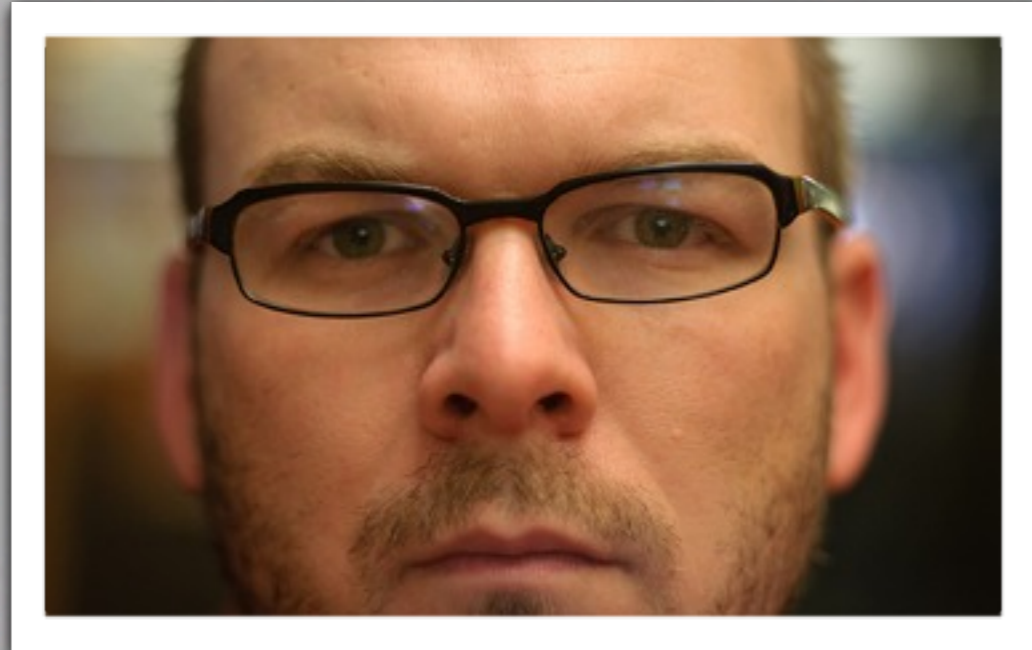
www.mikechambers.com

twitter.com/mesh

mesh@adobe.com