CROSS PLATFORM DEVELOPMENT

using the EverythingFlex AIR Library

by Rich Tretola

360|Flex Atlanta

Featured On:
O'REILLY
InsideRIA

ABOUT ME

Adobe COMMUNITY EXPERTS

HERFF JONES

SPEAKER
MAX
CONNECT. DISCOVER. INSPIRE.
Adobe

Sun microsystems
Certified Professional
JAVA™ TECHNOLOGY

<mx:EverythingFlex/>
by Rich Tretola

TOPICS

- Windows
- NativeMenus
- Dock/SystemTray Icons & Menus
- ContextMenus
- Alerts
- SuperWindow
- EverythingFlexAIR1.swc

WINDOWS

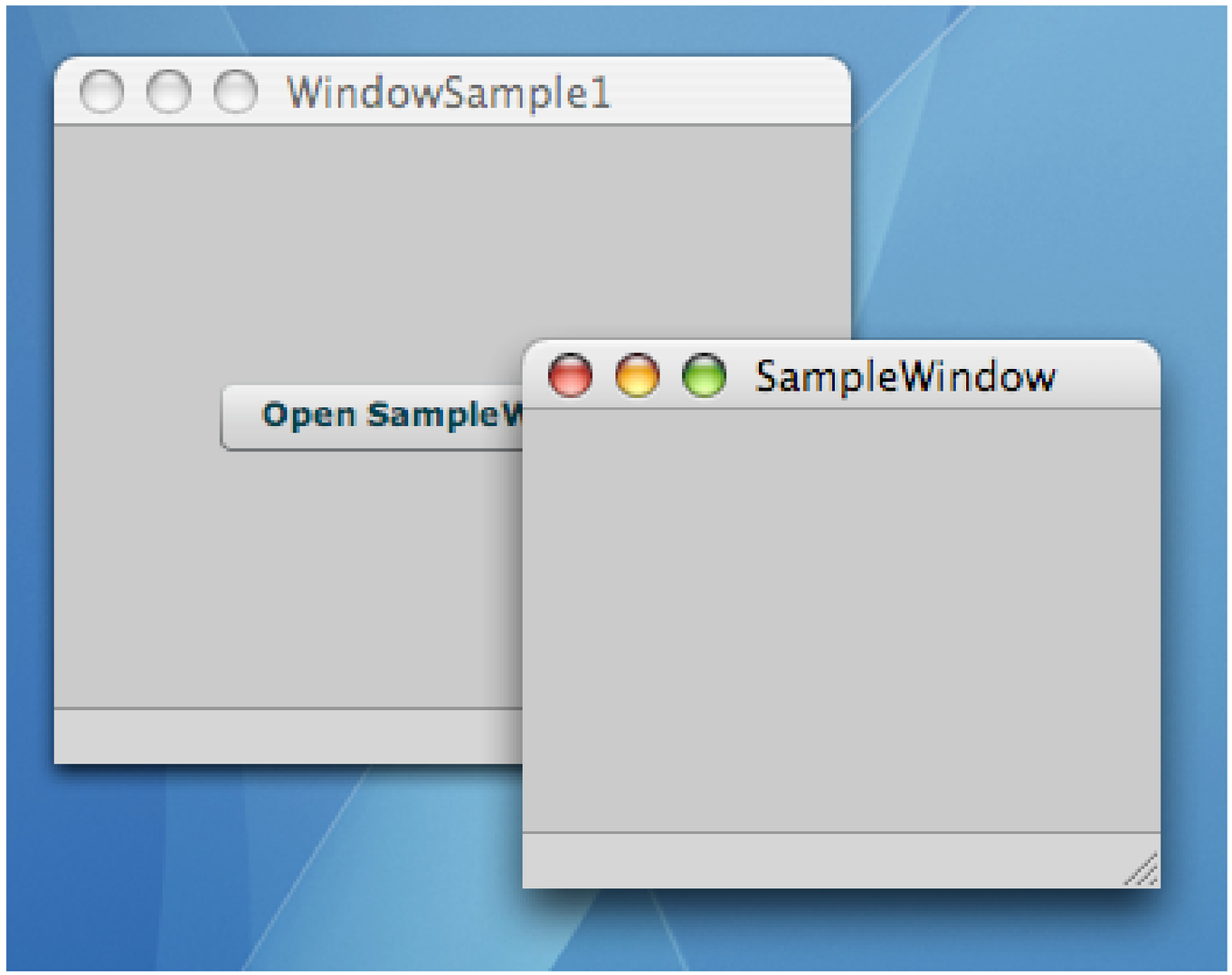
Windows have been a part of traditional desktop applications for many years. If you have ever opened a help window or a preferences window of a desktop application, you have most likely experienced a multi-window application.

AIR has given us the ability to create windows as part of AIR desktop applications as well. Each window is a fully-fledged system window and can act independently of the additional windows; it and can even persist after the main application window has been closed.

CREATE A WINDOW

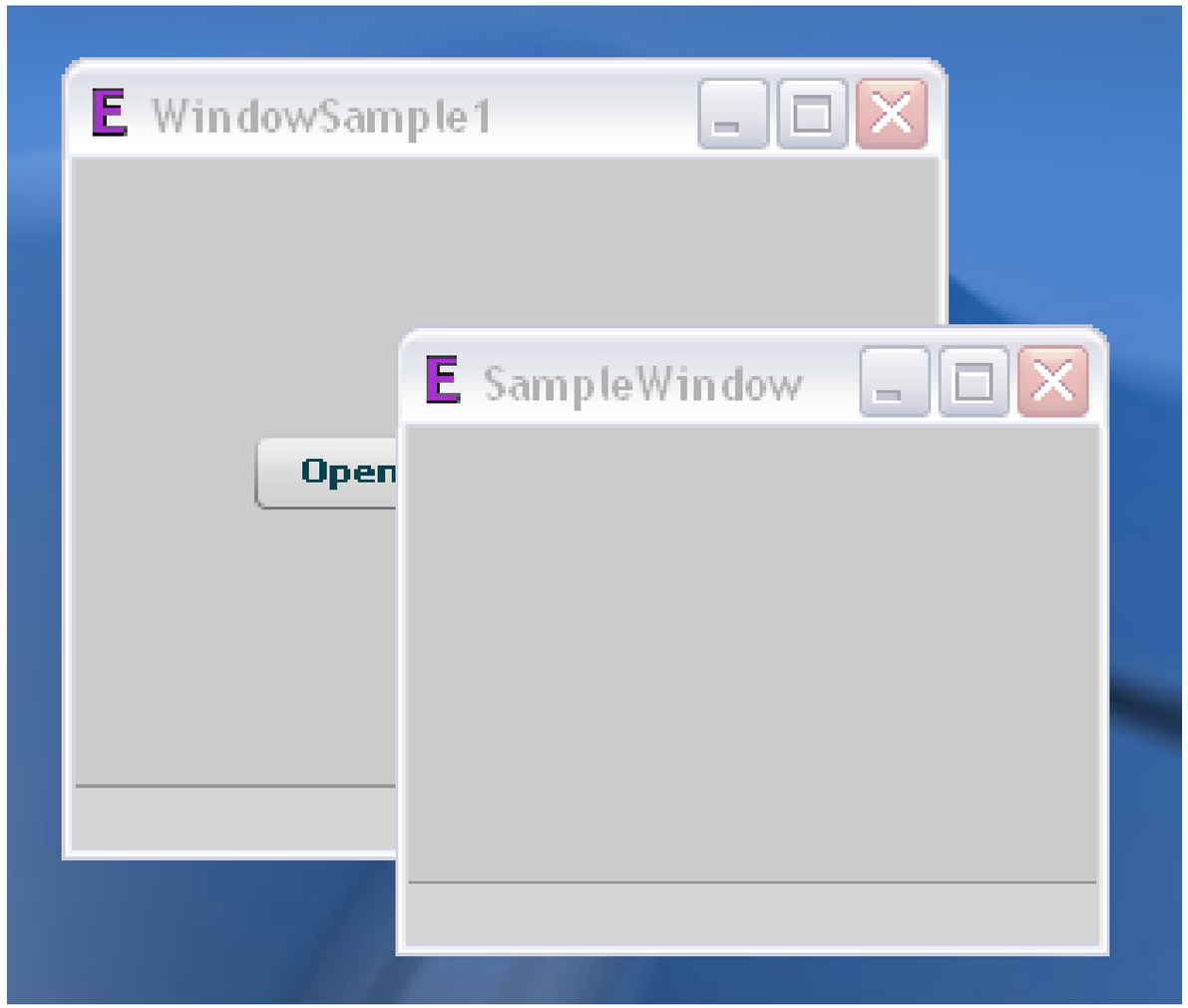
```
private function openSampleWindow():void{
    // create the new window and add some properties
        var win:Window = new Window();
        win.width = 200;
        win.height = 150;
        win.title = "SampleWindow";
        win.open();
}
```

RESULTS ON MAC



WindowSample1

RESULTS ON WINDOWS



WindowSample1

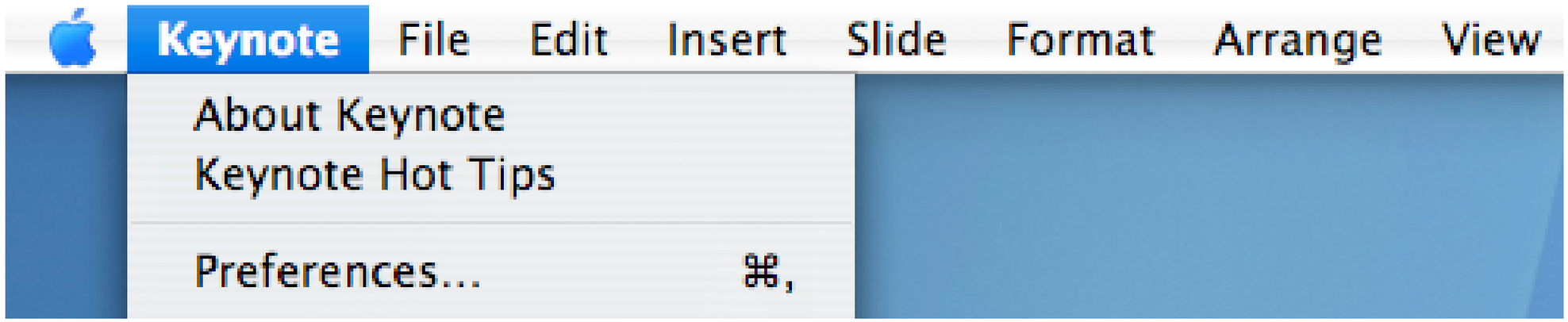
NATIVEMENUS

AIR has also provided the ability to integrate menus into your applications.

However, the implementation is different depending on which operating system your application is currently running on.

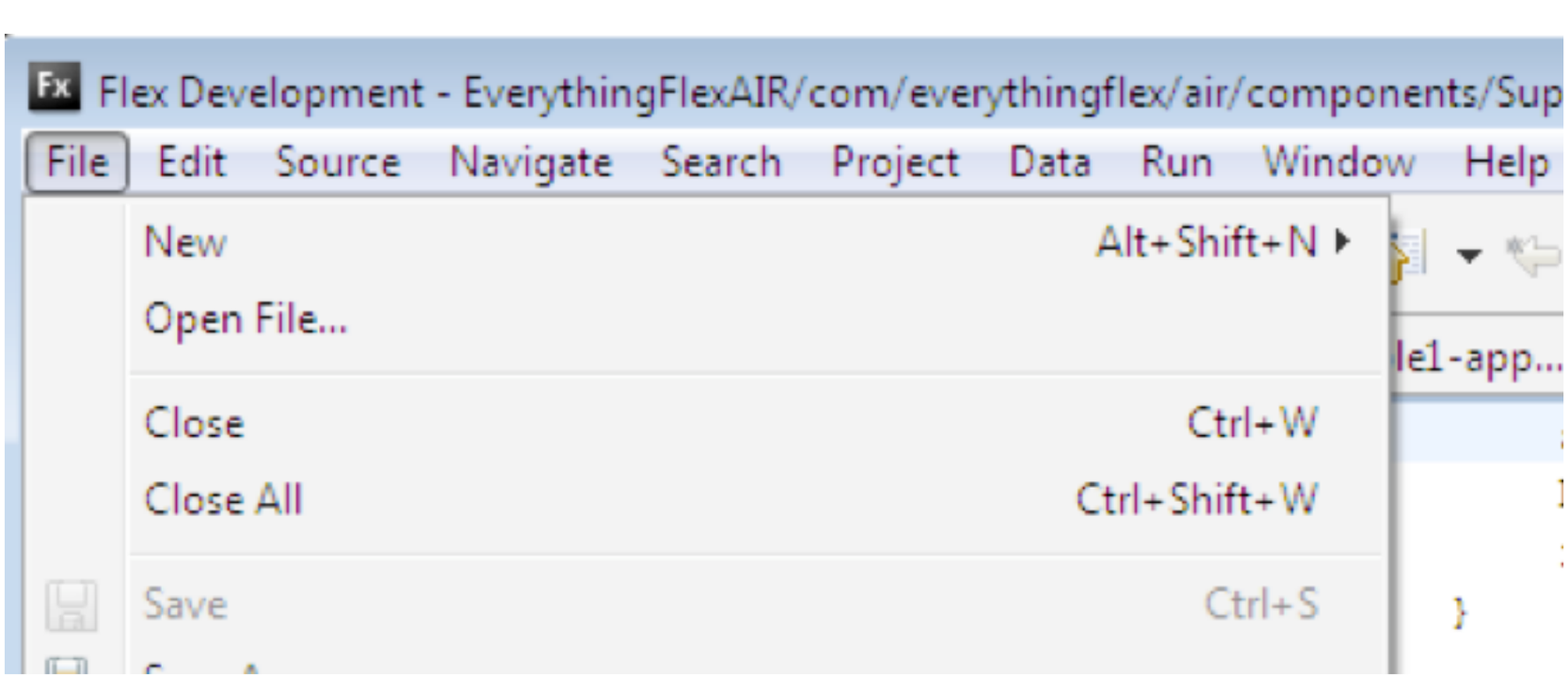
MAC OSX

Applications that run on Mac OS X like the one I am currently using have the application menu in an upper control bar.



WINDOWS XP, VISTA

Windows 2000, XP, Vista, etc include menus within the application window as shown in the screen shot of Flex Builder below.



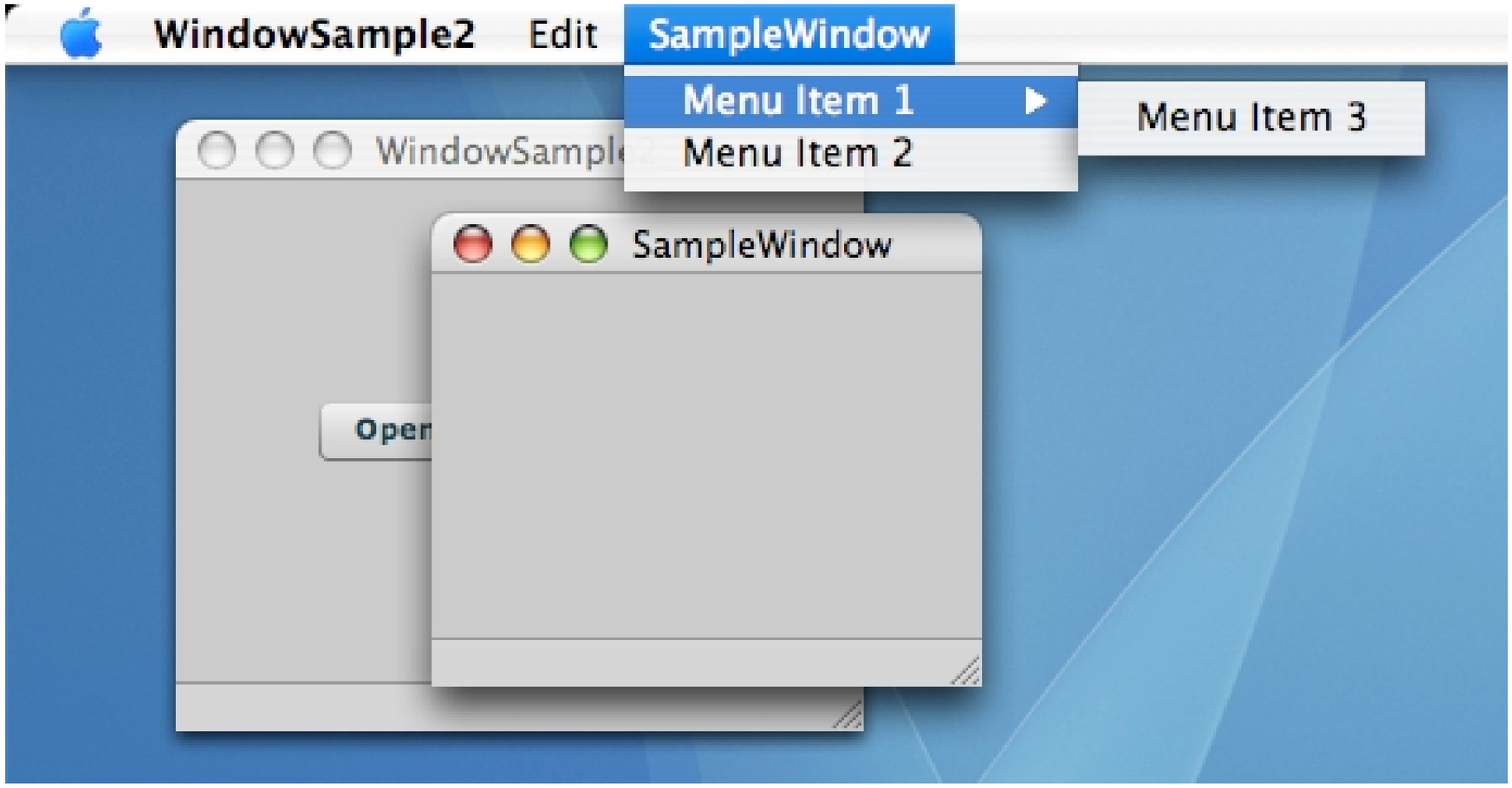
CREATE A NATIVEMENU

```
// create main menu
var nativeMenu:NativeMenu = new NativeMenu();
// create a few menu items
var menuItem1:NativeMenuItem = new NativeMenuItem("Menu Item 1");
var menuItem2:NativeMenuItem = new NativeMenuItem("Menu Item 2");
// add menu items
nativeMenu.addItem(menuItem1);
nativeMenu.addItem(menuItem2);
```

NATIVEMENU ON MAC

```
// if supported, add a menuItem to application
if(NativeApplication.supportsMenu){
var menuItem:NativeMenuItem = new NativeMenuItem("Menu Name");
      menuItem.submenu = nativeMenu;
NativeApplication.nativeApplication.menu.addItem(menuItem);
}
```

RESULTS ON MAC

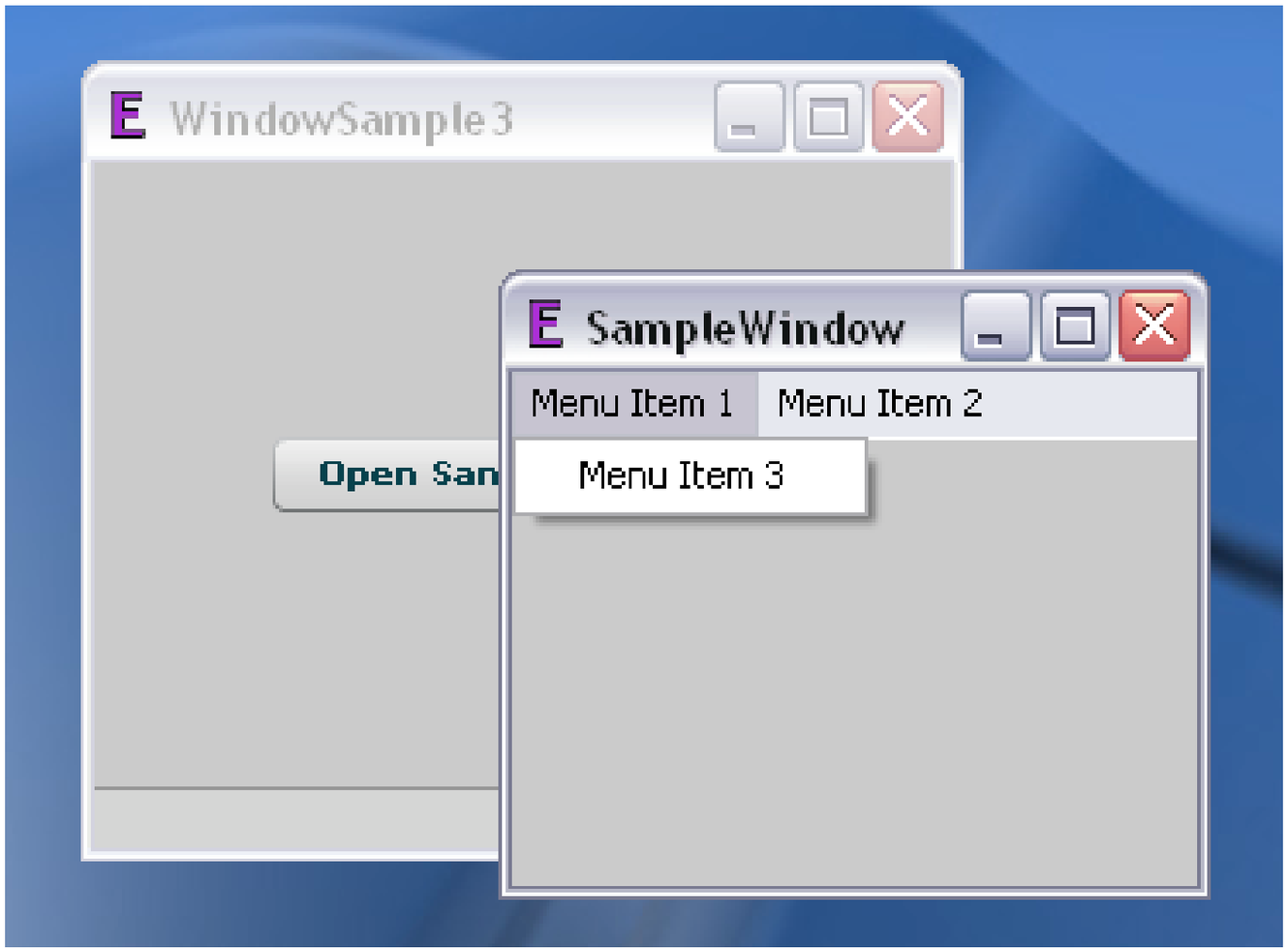


WindowSample2

NATIVEMENU ON WINDOWS

```
// if supported, add a nativeMenu to win
if(NativeWindow.supportsMenu){
    win.nativeWindow.menu = nativeMenu;
}
```

RESULTS ON WINDOWS



WindowSample3

DOCK & SYSTEMTRAY ICONS

The Dock icon on the Mac and the icons that appear in the SystemTray on Windows machines offer insight and alerts to the user of changes within the application.

SET A DOCK OR SYSTEMTRAY ICON

```
[Embed(source="assets/e16.png")]
private var Icon16:Class;
private var bitmap16:Bitmap = new Icon16();

[Embed(source="assets/e32.png")]
private var Icon32:Class;
private var bitmap32:Bitmap = new Icon32();

[Embed(source="assets/e48.png")]
private var Icon48:Class;
private var bitmap48:Bitmap = new Icon48();

[Embed(source="assets/e128.png")]
private var Icon128:Class;
private var bitmap128:Bitmap = new Icon128();
```

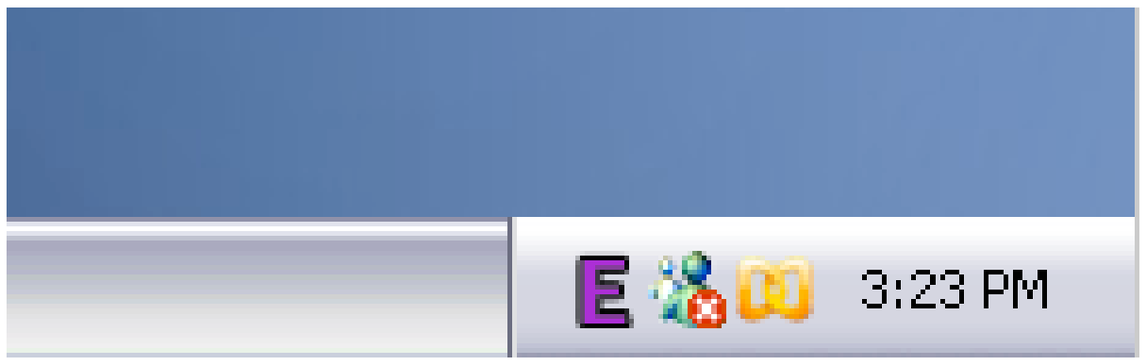
SET A DOCK OR SYSTEMTRAY ICON

```
private function setDockIcon():void{
var icons:Array = [bitmap16.bitmapData,
                   bitmap32.bitmapData,
                      bitmap48.bitmapData,
                      bitmap128.bitmapData]
      NativeApplication.nativeApplication.icon.bitmaps =
icons;
}
```

RESULTS

Mac

Windows



DockTrayIcon1

DOCK & SYSTEMTRAY TOOLTIPS & MENUS

Only the Windows SystemTray supports icon tooltips.

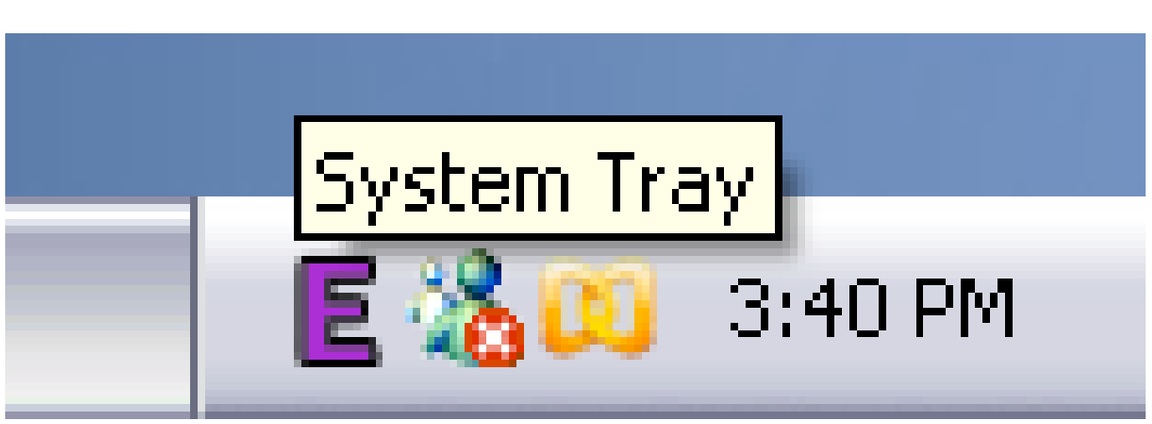
Both Dock & SystemTray icons support menus being attached to them for click and hold on Mac and right click on Windows.

SET A SYSTEMTRAY ICON TOOLTIP

```
private function setSystemTrayToolTip():void{
if(NativeApplication.supportsSystemTrayIcon){
      SystemTrayIcon(NativeApplication.nativeApplication.icon).
      tooltip = "System Tray";
     }
}
```

RESULTS

Windows



DockTrayIcon2

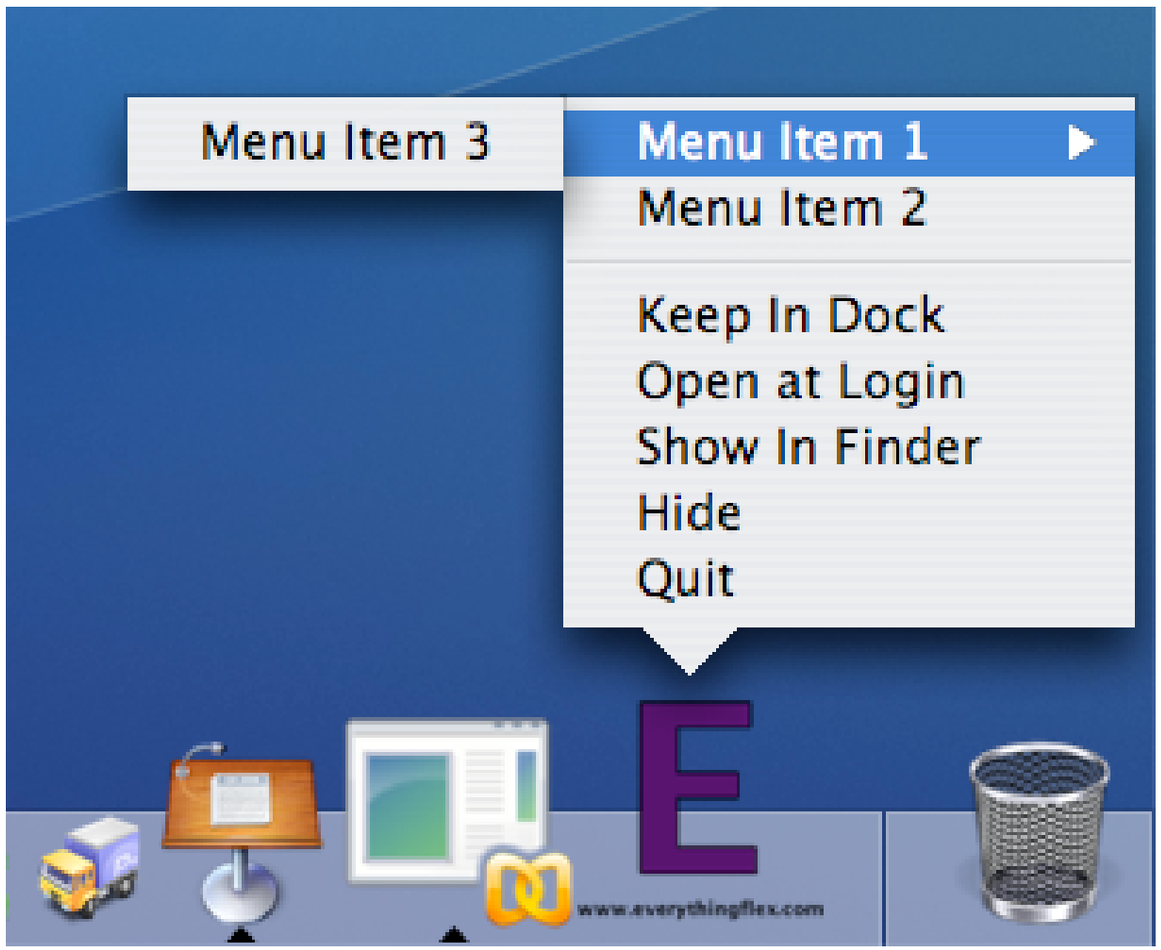
SET A DOCK OR SYSTEMTRAY MENU

```
// check for support of DockIcon
if(NativeApplication.supportsDockIcon) {
DockIcon(NativeApplication.nativeApplication.icon).
menu = nativeMenu;
}
// check for support of SystemTrayIcon
if(NativeApplication.supportsSystemTrayIcon){

SystemTrayIcon(NativeApplication.nativeApplication.icon).
menu = nativeMenu;
}
```

RESULTS

Mac



DockTrayIcon3

RESULTS

Windows

DockTrayIcon3

CONTEXT MENUS

AIR gives developers the ability to create custom Context menus which are right-click on Windows and ctrl-click on Mac.

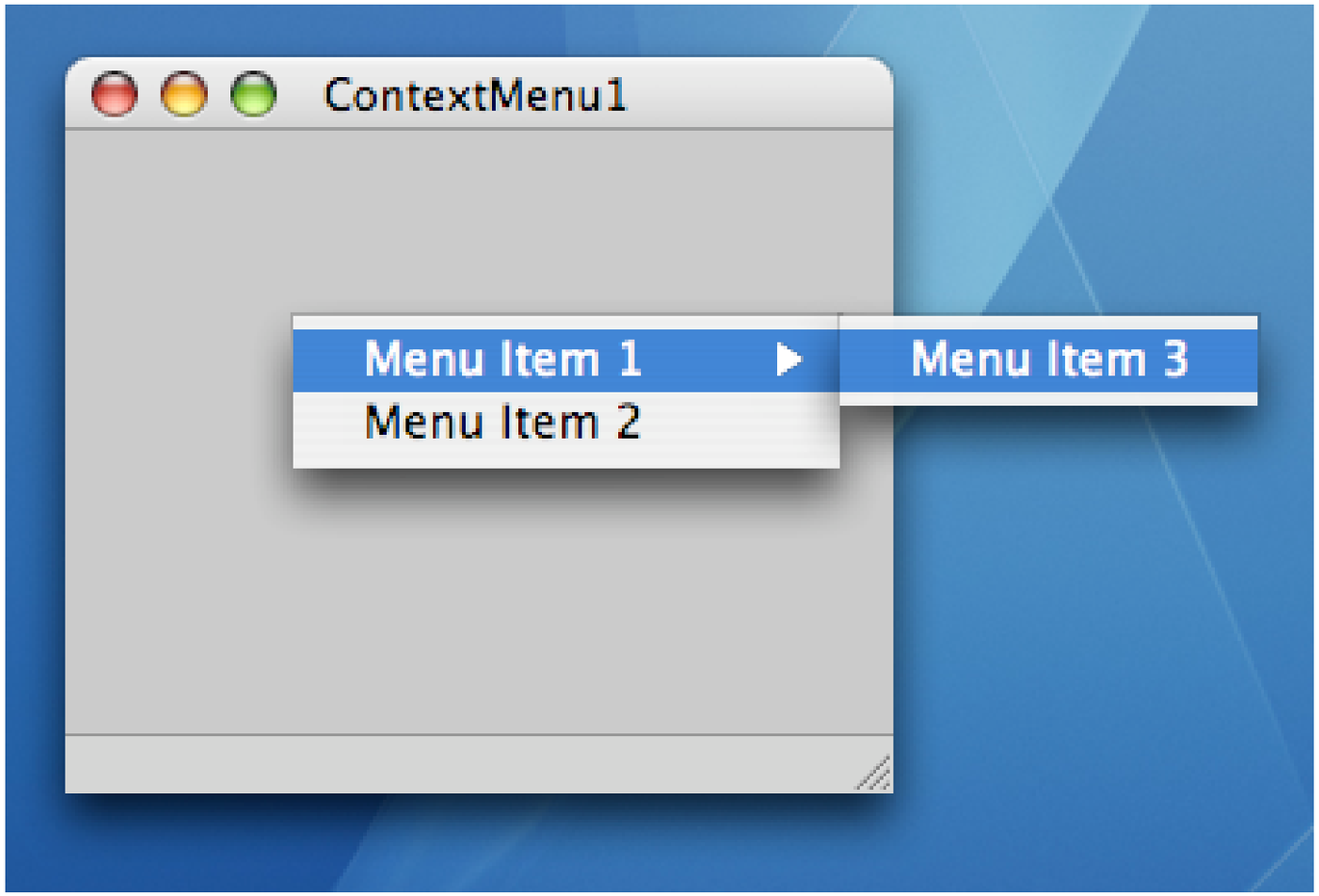
Context menus are created in the same manner as NativeMenus.

CREATE A CONTEXT MENU

```
// create main menu
var nativeMenu:NativeMenu = new NativeMenu();
// create a few menu items
var menuItem1:NativeMenuItem = new NativeMenuItem("Menu Item 1");
var menuItem2:NativeMenuItem = new NativeMenuItem("Menu Item 2");
// add menu items
nativeMenu.addItem(menuItem1);
nativeMenu.addItem(menuItem2);
// create a submenu
var subMenu:NativeMenu = new NativeMenu();
// create a submenu item
var menuItem3:NativeMenuItem = new NativeMenuItem("Menu Item 3");
// add item to submenu
subMenu.addItem(menuItem3);
// set the submenu to menuItem1
menuItem1.submenu = subMenu;
// add Context menu to main application window
this.contextMenu = nativeMenu;
```

RESULTS

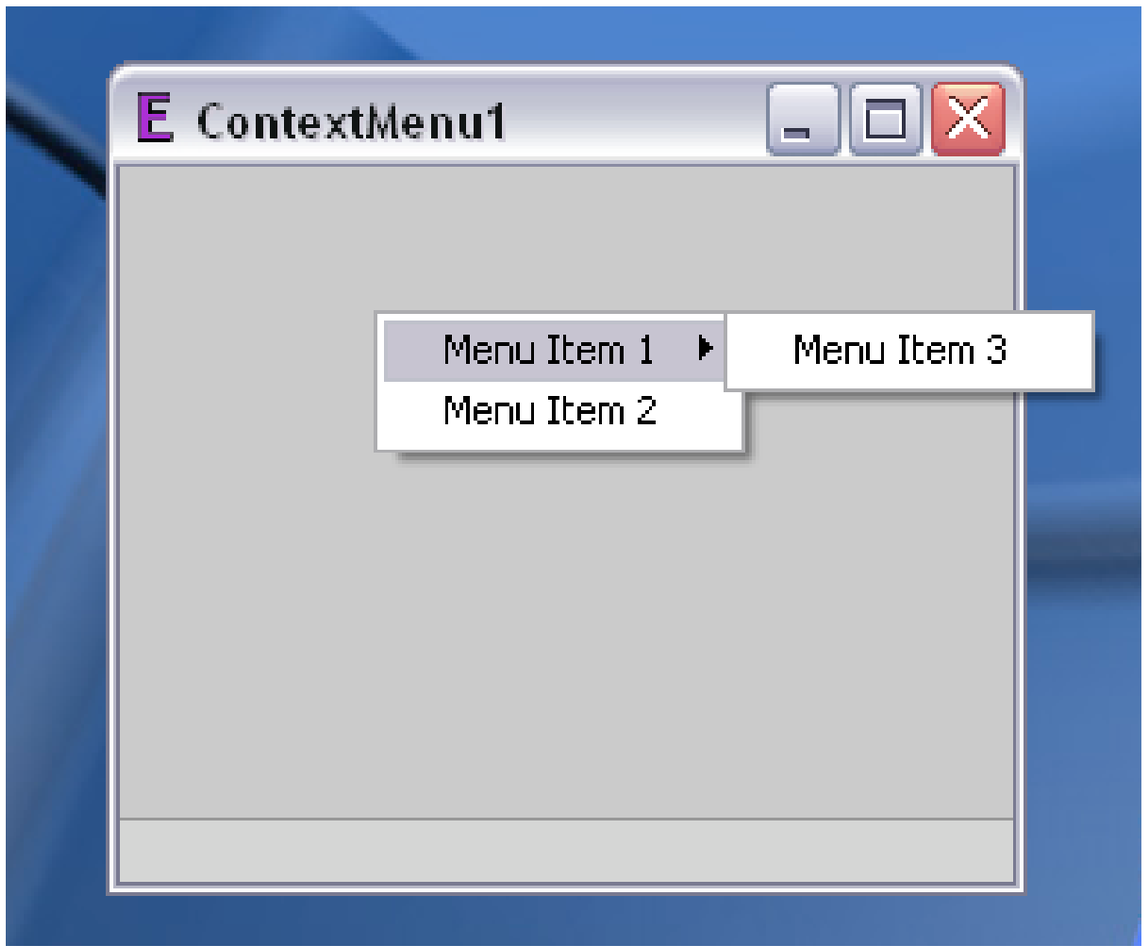
Mac



ContextMenu1

RESULTS

Windows



ContextMenu1

CONTEXT MENUS

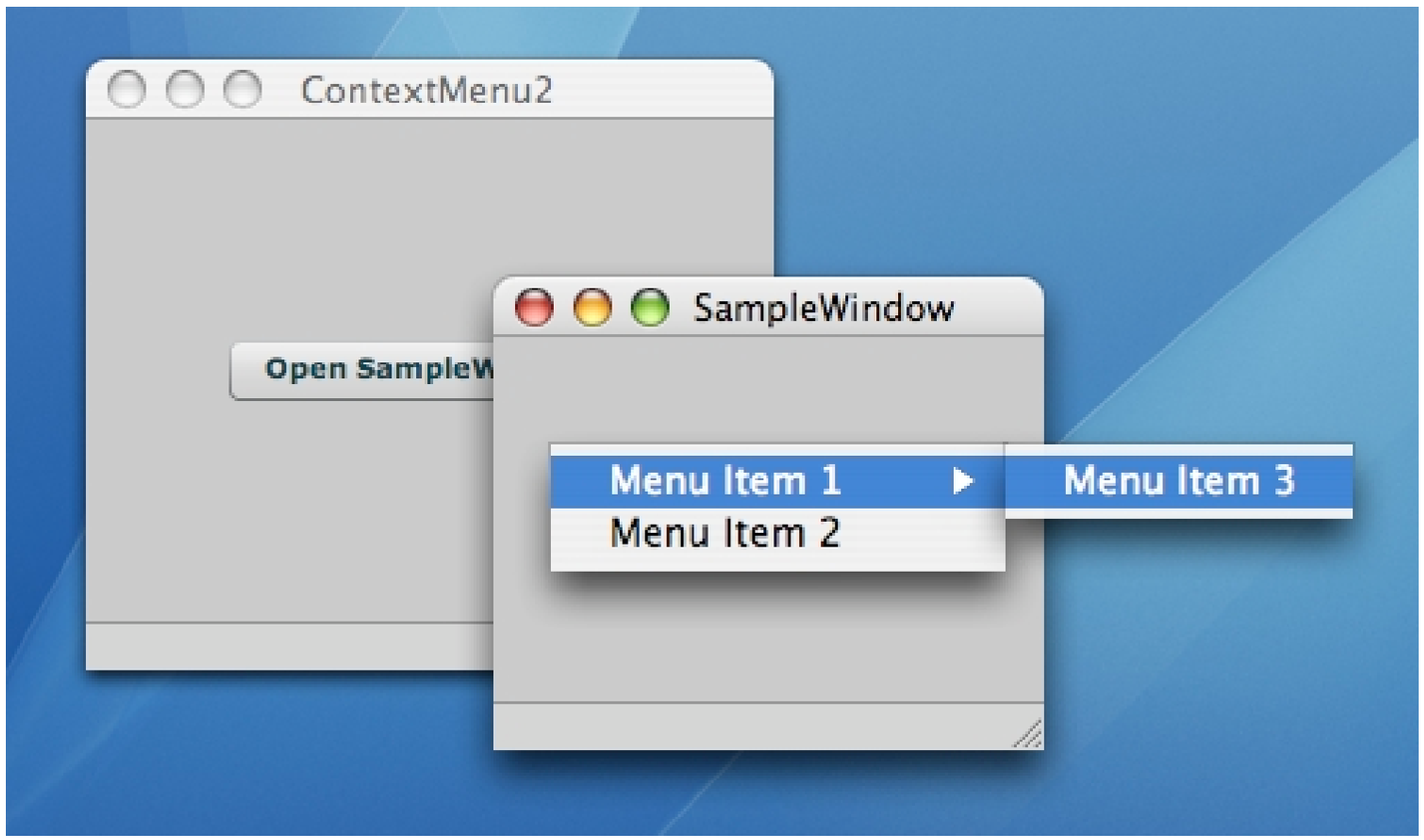
Context Menus can also be assigned per Window within AIR applications.

CONTEXT MENUS

```
var win:Window = new Window();
    win.width = 200;
    win.height = 150;
    win.title = "SampleWindow";
    win.open();
    // add context menu to win
    win.contextMenu = nativeMenu;
```

RESULTS

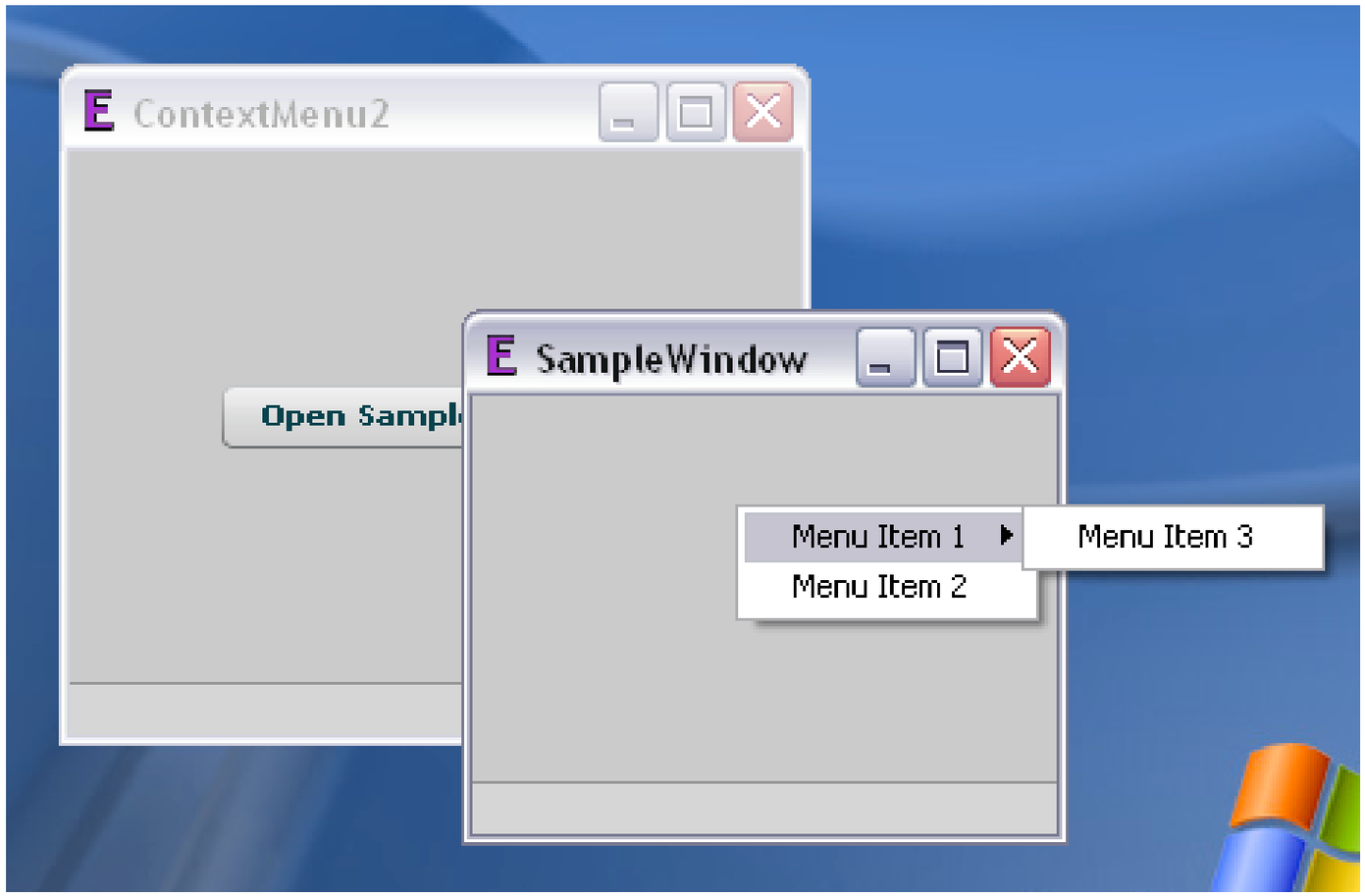
Mac



ContextMenu2

RESULTS

Windows



ContextMenu2

ALERTS

There is a concept of alerting the user that the application needs attention with AIR on Mac by bouncing the dock icon.

On Windows, you can set a window in the task bar to blink.

DOCKICON BOUNCE

```
if(NativeApplication.supportsDockIcon){

DockIcon(NativeApplication.nativeApplication.icon).
bounce(NotificationType.CRITICAL);
}
```

NotificationType can be CRITICAL which will bounce until a user interaction occurs or INFORMATIONAL which will simply bounce once.

DockBounce

WINDOW NOTIFY

```
if(NativeWindow.supportsNotification){

this.nativeWindow.notifyUser(NotificationType.CRITICAL);
}
```

NotificationType can be CRITICAL which will blink until a user interaction occurs or INFORMATIONAL which will simply show a highlight on the window in the task bar.

WindowNotify

So?

Wouldn't it be great if all of the functionality that I have just shown you was able to be implemented with just a few lines of code?

GOOD NEWS!

The SuperWindow component part of the EverythingFlexAIR1.swc will let you manage menus, icons, icon menus, alerts, and context menus. Plus even more!!

WHAT IS SUPERWINDOW?

The SuperWindow component was originally created to solve the job of managing NativeMenus.

As previously demonstrated NativeMenus function differently on Mac vs Windows.

MANAGING NATIVEMENUS

- SuperWindow on Mac can operate in either single or multi menu mode.

- The static property MULTI_MENU_MODE can be set to true or false. It is false by default.

- This is a per application setting.

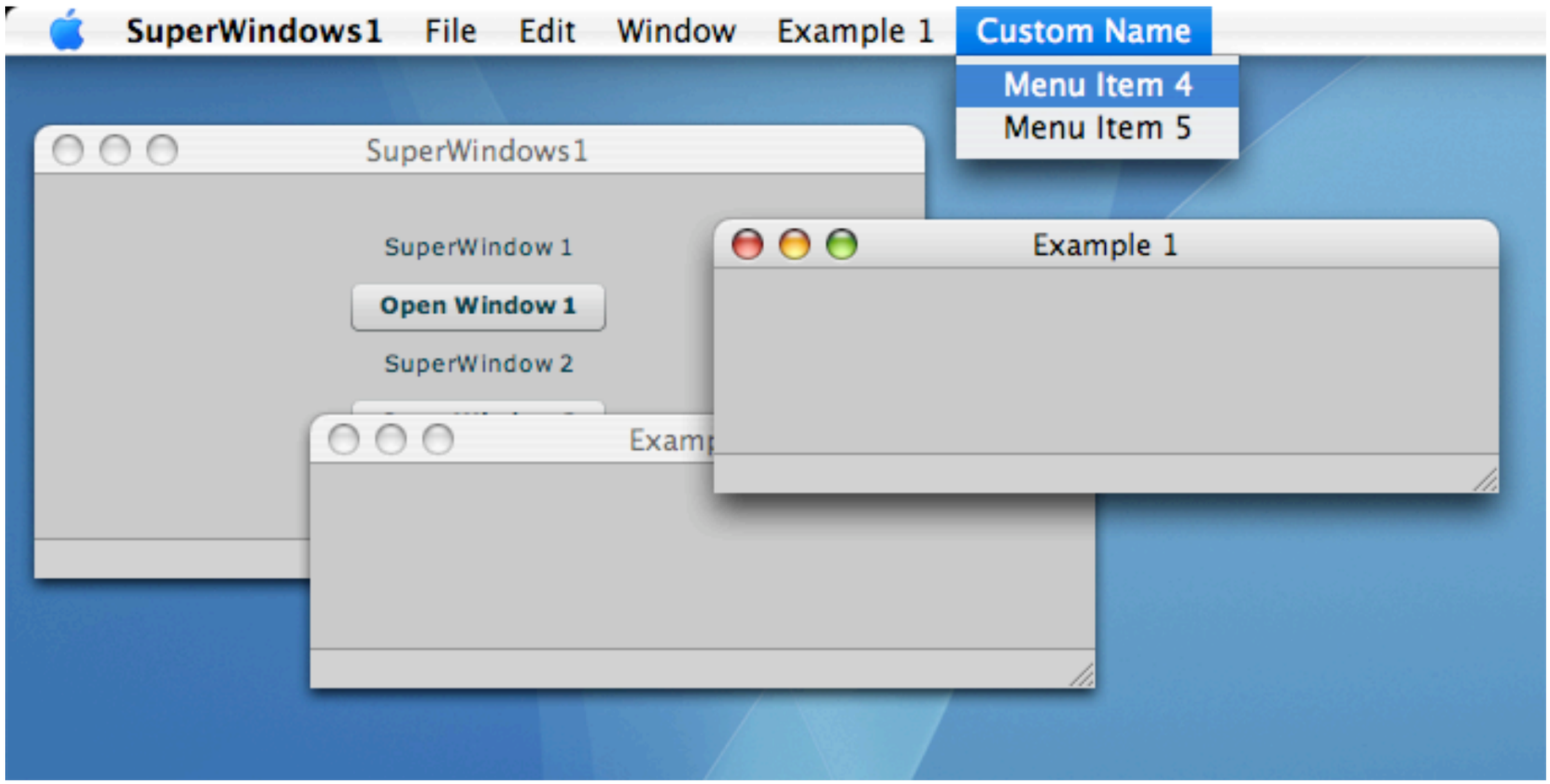
MENU PROPERTIES

- The NativeMenu is set by setting the SuperWindow's nm property.
- The menu will either be created with the name of the accompanying Window's title property or can accept a custom name by using the menuName property of the SuperWindow class.

MULTI_MENU_MODE = "TRUE"

When MULTI_MENU_MODE=”true” the application will be in multi menu mode. In this mode a new menu will be added to the right of the default AIR menus and will only be removed when the accompanying Window is closed.

MULTI_MENU_MODE = "TRUE"

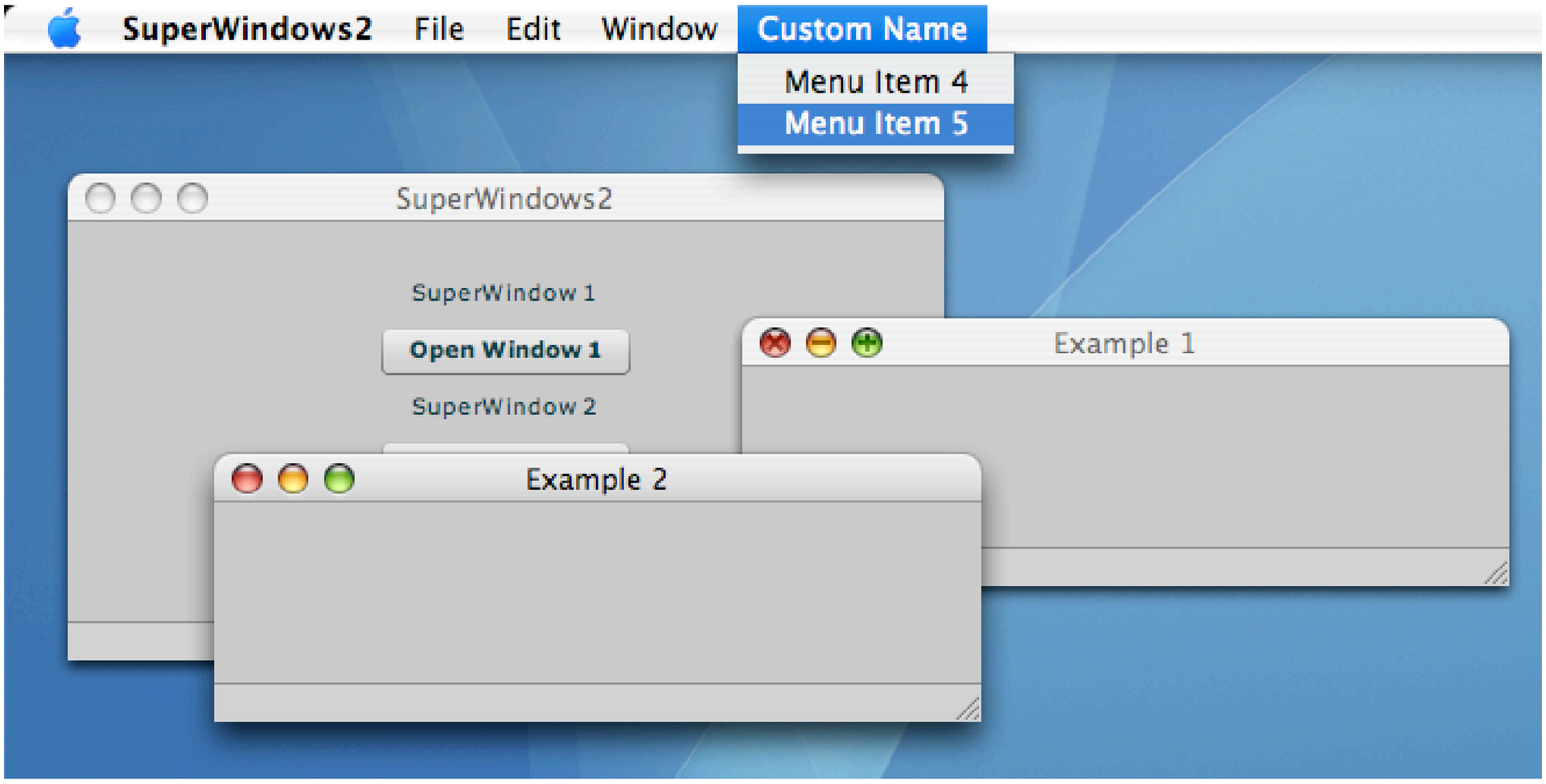


SuperWindows1

MULTI_MENU_MODE = "FALSE"

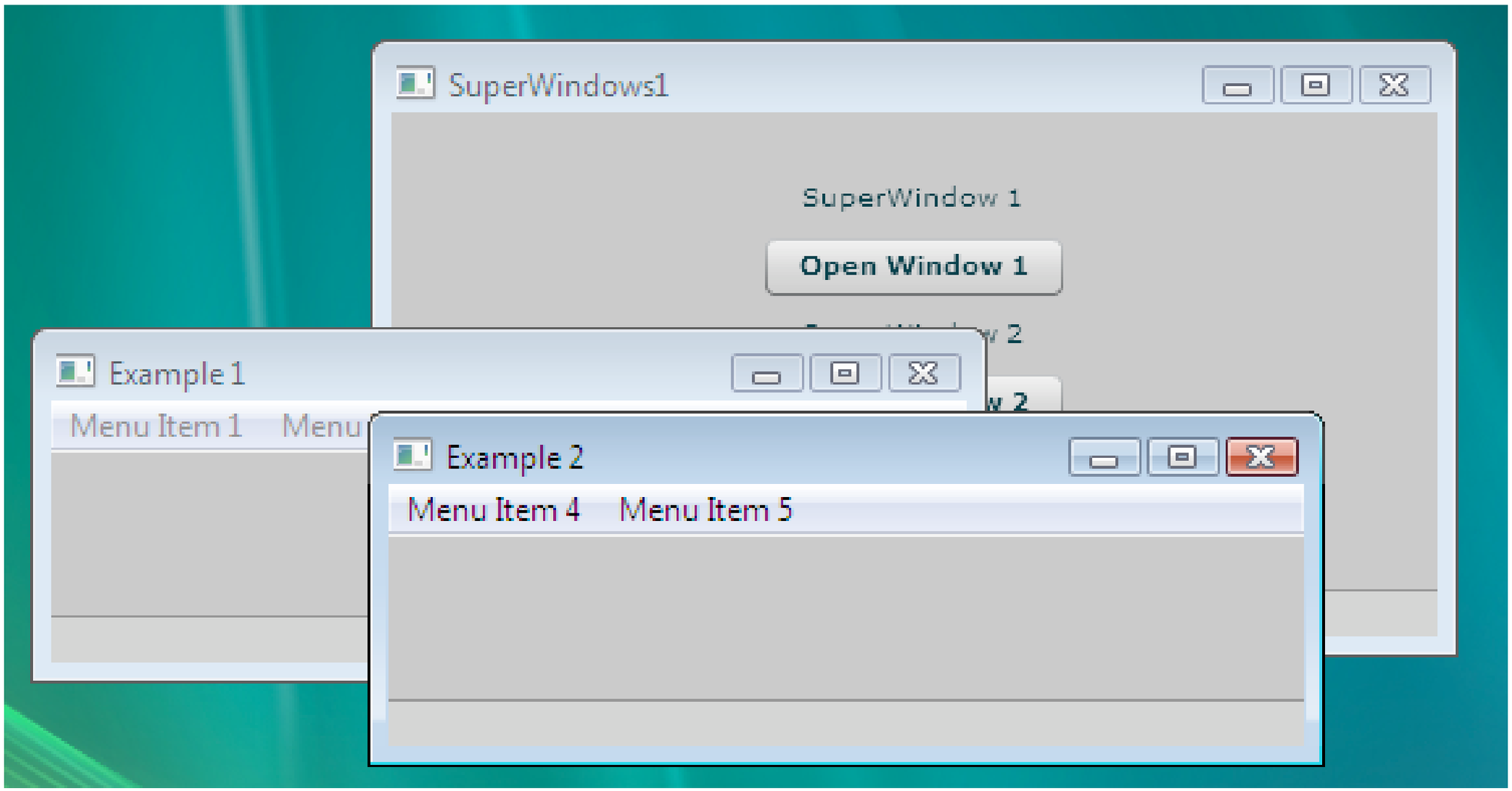
When MULTI_MENU_MODE=”false” the application will be in single menu mode. In this mode one menu will be added to the right of the default AIR menus and will be swapped each time a Window gains focus.

MULTI_MENU_MODE = "FALSE"



SuperWindows2

ON WINDOWS



SuperWindows1

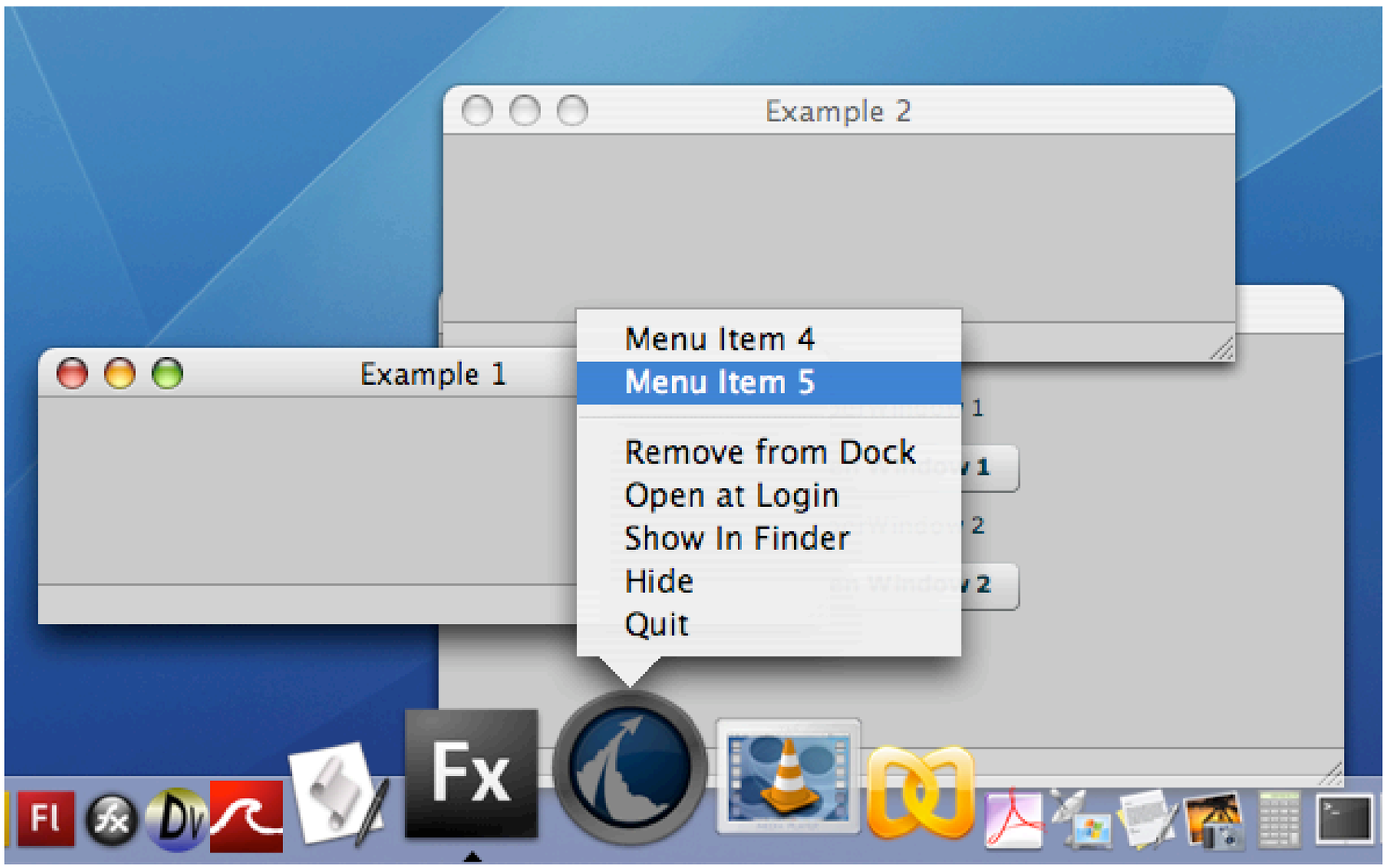
DOCK SYSTEMTRAY MENUS

- SuperWindow allows for the reuse of the supplied NativeMenu as the Dock or SystemTray menu.

- To accomplish this, you simply set the static SHOW_SYSDOCK_MENU property to true. It is false by default.

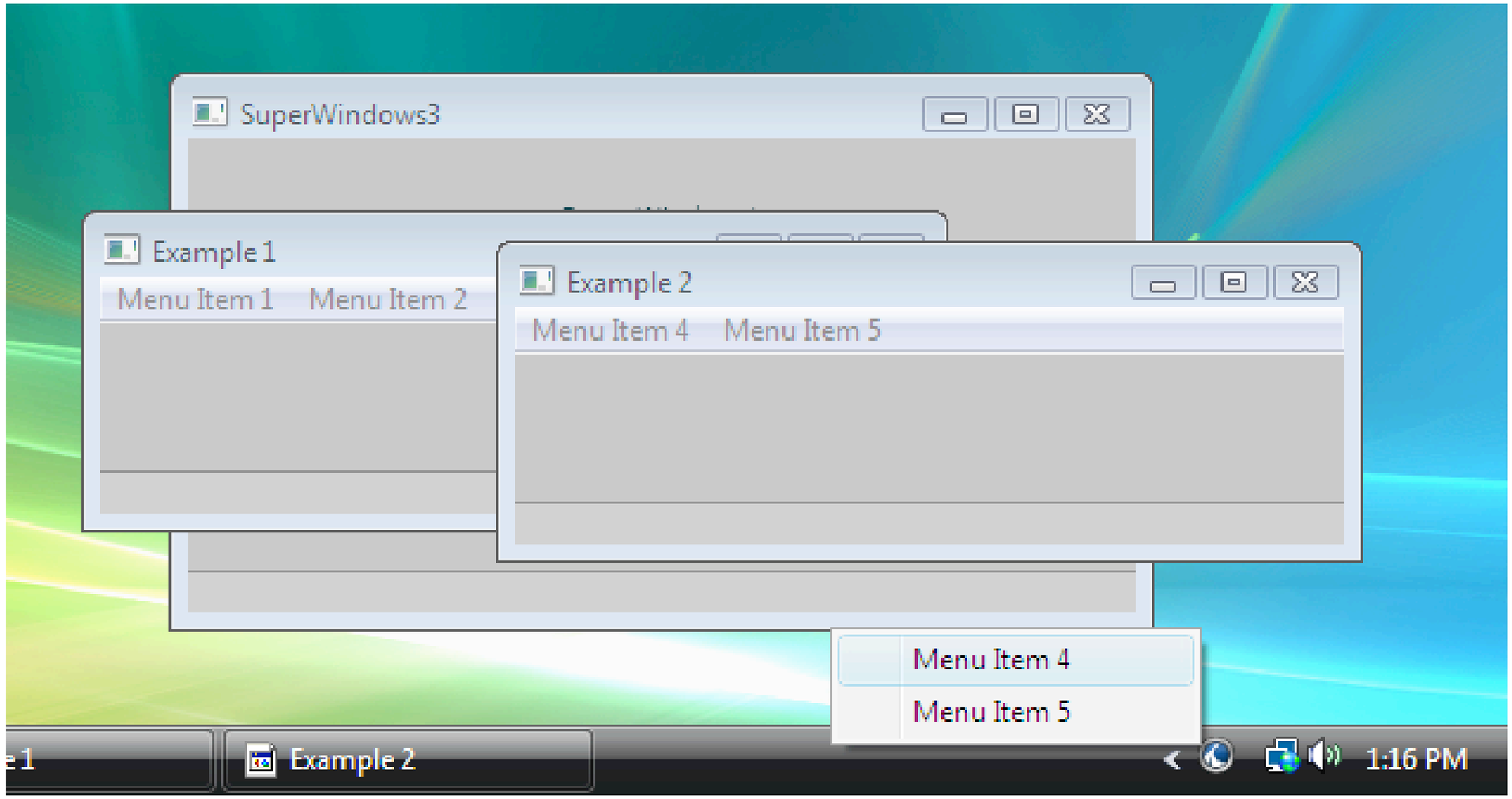
- This is a per application setting.

SHOW_SYSDOCK_MENU ="TRUE"



SuperWindows3

SHOW_SYSDOCK_MENU ="TRUE"



SuperWindows3

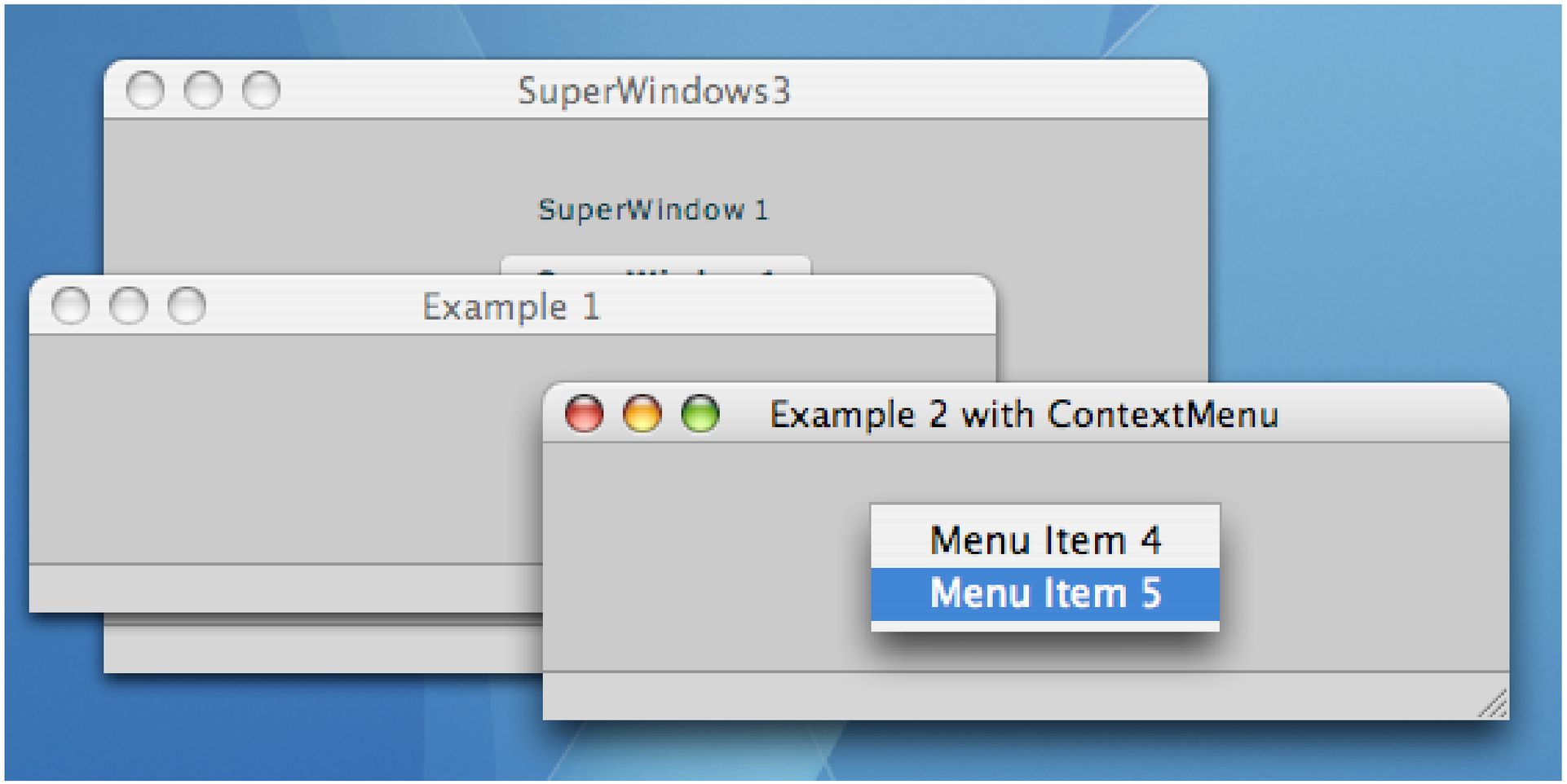
CONTEXT MENUS

- SuperWindow also allows for a simple reuse of the supplied NativeMenu as the window's ContextMenu.

- To accomplish this, you simply set the window's showContextMenu property to true.

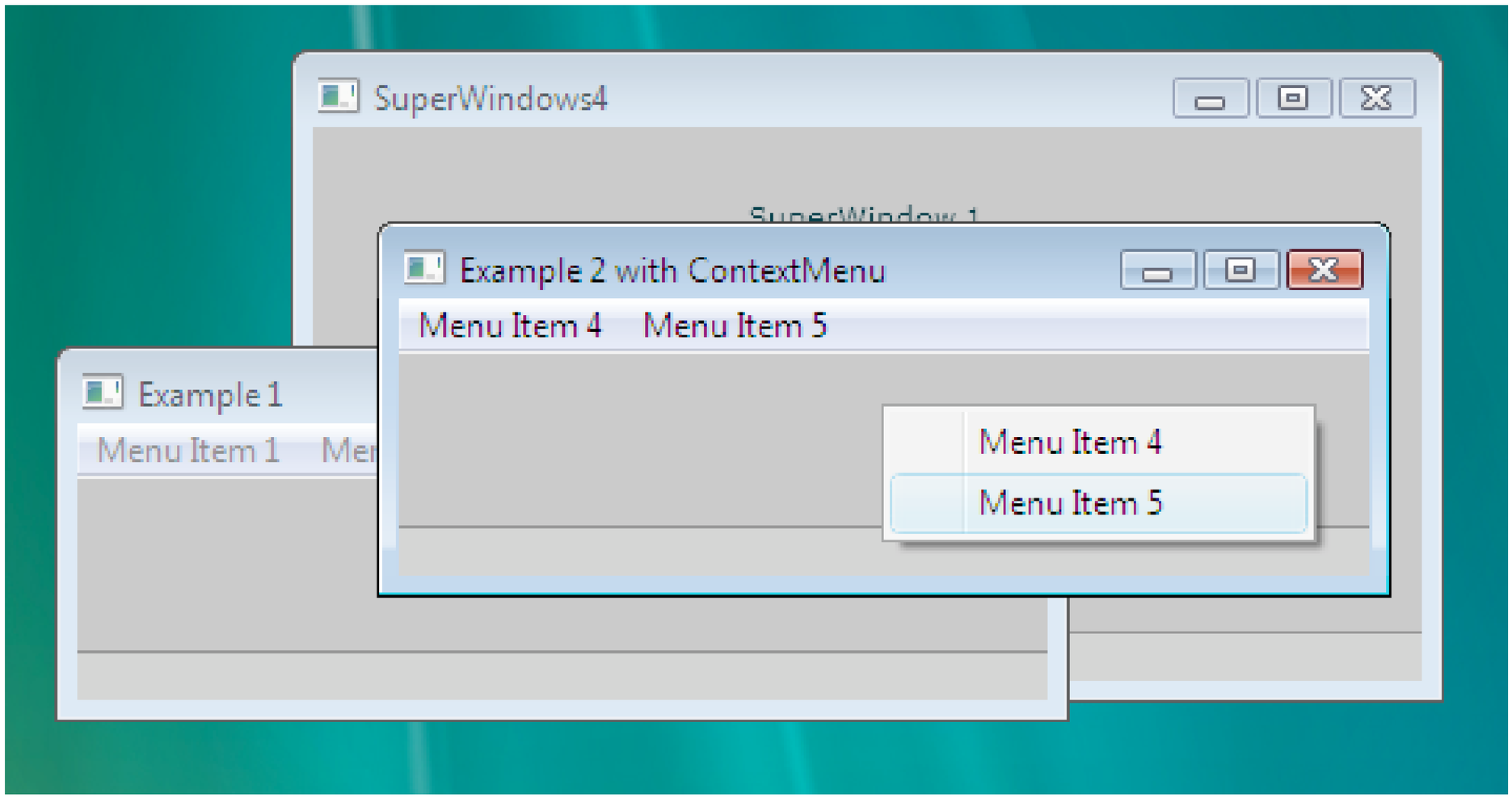
- This is a per window setting.

SHOWCONTEXTMENU="TRUE"



SuperWindows4

SHOWCONTEXTMENU="TRUE"



SuperWindows4

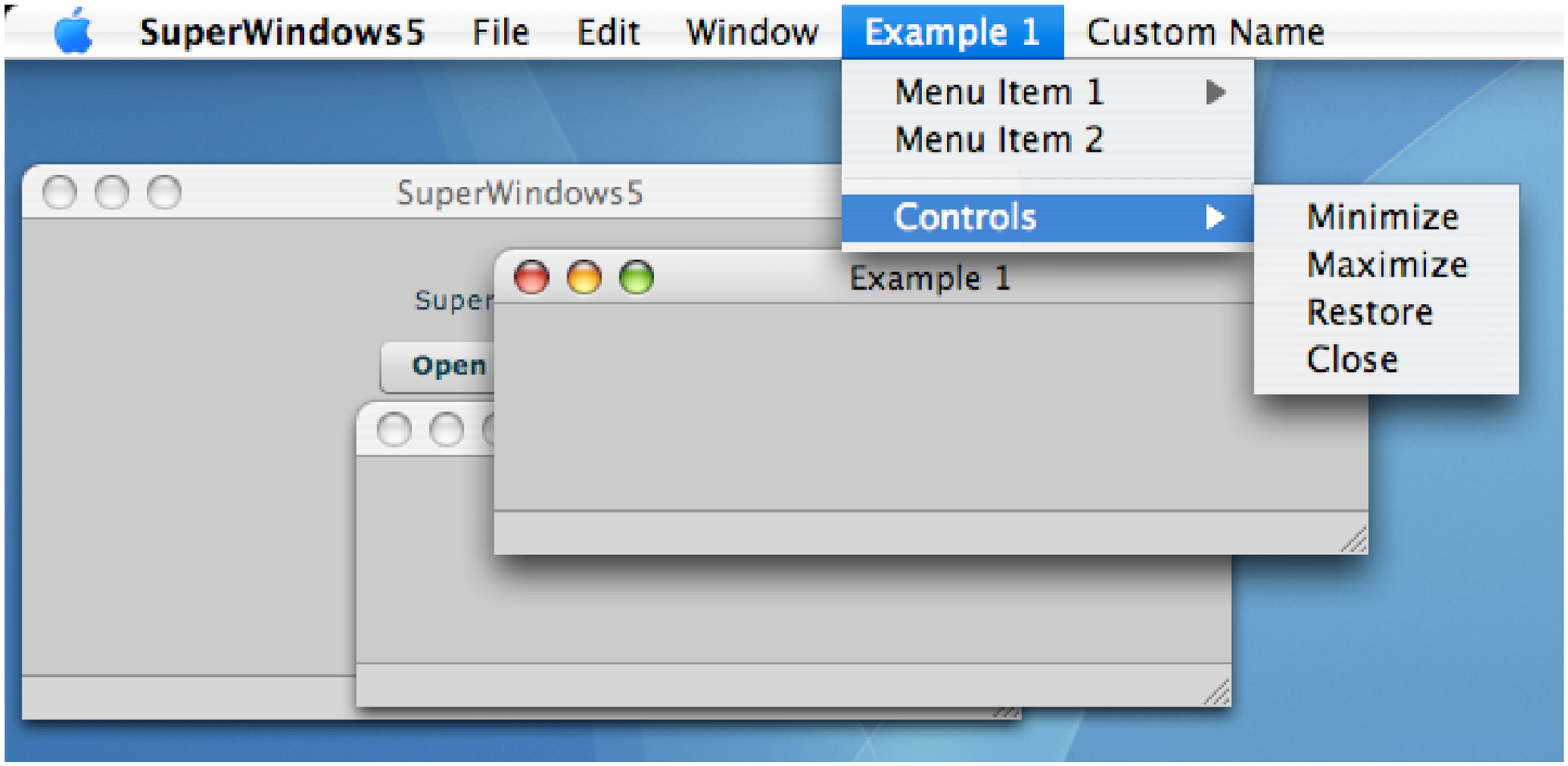
ADD WINDOW CONTROLS

- SuperWindow will also allow you to automatically add window controls to your NativeMenus which will then show in all of the locations previously covered.

- To accomplish this, you simply set the window's addWindowControls property to true.

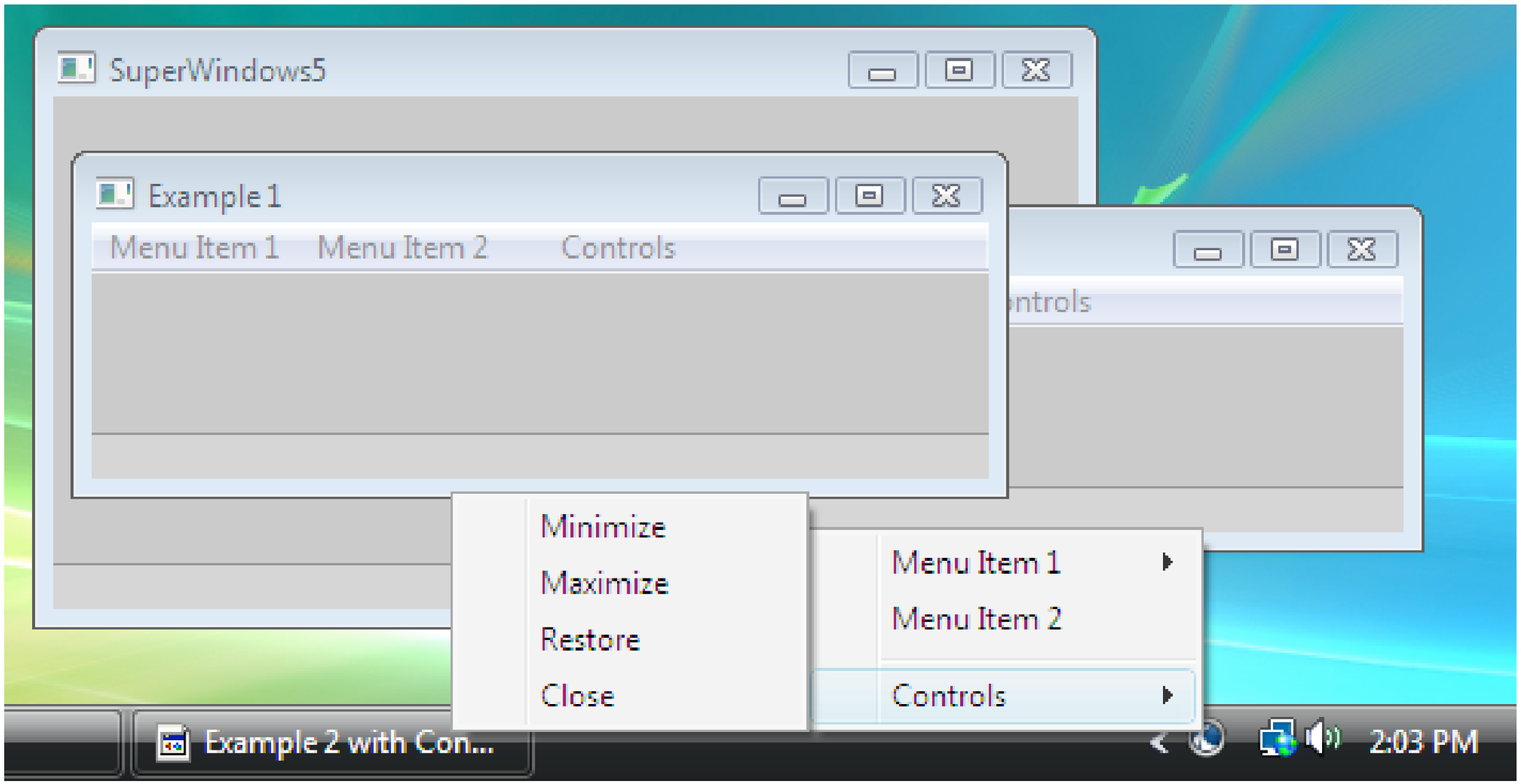
- This is a per window setting.

ADDWINDOWCONTROLS ="TRUE"



SuperWindows5

ADDWINDOWCONTROLS ="TRUE"

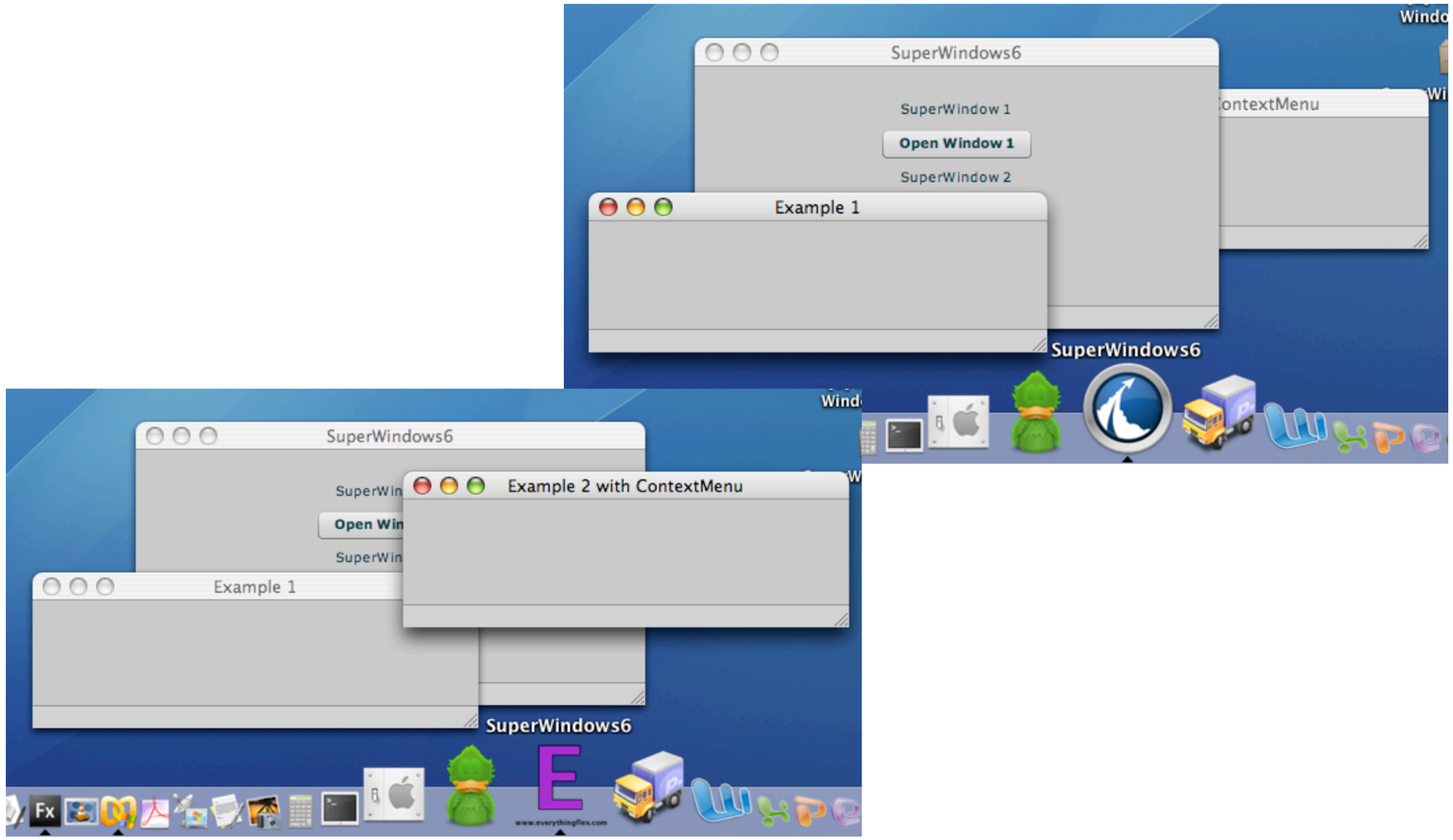


SuperWindows5

DOCK SYSTEM TRAY ICONS

- SuperWindow supports per window Dock or SystemTray Icons.
- Simply set an Array of bitmapData to the sysDockIconBitmaps property and SuperWindow will construct and manage the Dock or SystemTray Icon setting the icon to the current window.
- This is a per window setting.

SYSDOCKICONBITMAPS



SuperWindows6

ALERTS METHODS

SuperWindow supports 3 Alert methods.

- Native: the bounce on Mac and the taskbar Window flash on Windows
- Icon: the icon will change from its current view to a negative view of itself or a supplied altered icon
- Toast: toast style windows are available as an alert method

Native is the default and the alerts are per Window

ALERT TYPES

SuperWindow supports both NotificationType.CRITICAL and NotificationType.INFORMATIONAL

The default is NotificationType.CRITICAL

NATIVE ALERTS

SuperWindow supports both NotificationType.CRITICAL and NotificationType.INFORMATIONAL

The default is NotificationType.CRITICAL

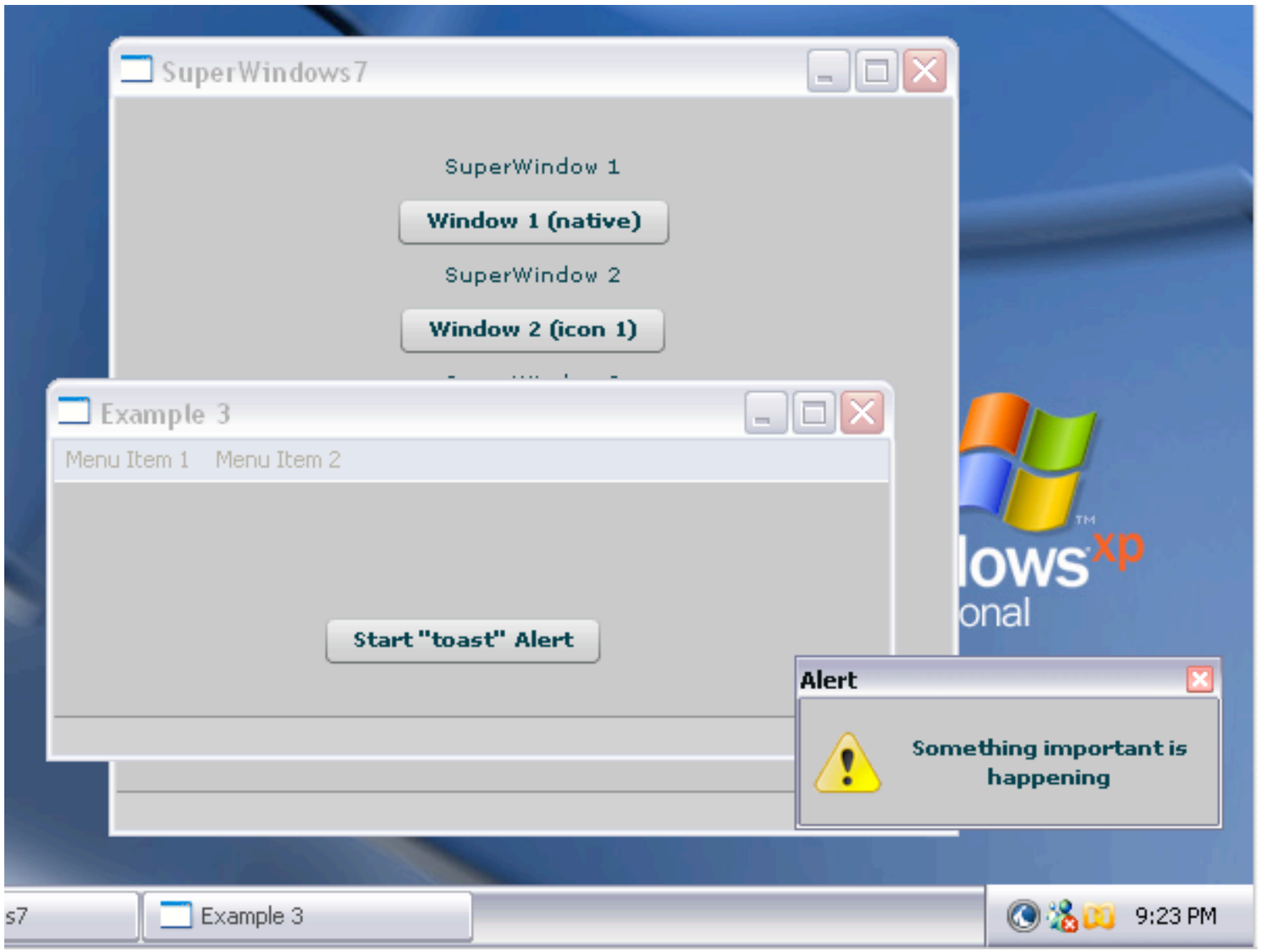
STARTALERT()

To start a SuperWindow alert, a call to the startAlert() method is necessary.

The startAlert() method accepts one argument of type String which is the message the will display as the Icon toolTip or the message within a toast alert.

Ex: startAlert(“Something is happening”);

TOAST ALERT



SuperWindows7

WHAT ELSE?

OK, so we have seen the SuperWindow component, but what else is included in the EverythingFlexAIR1.swc component library?

EVERYTHINGFLEXAIR1.SWC

- SuperWindow
- ContextWindow
- AlertWindow
- ConnectionManager
- UpdateManager

CONTEXTWINDOW

The ContextWindow component was created to easily add the “close”, “minimixe”, “maximize”, and ‘restore” functionality to AIR windows.

The functionality of the ContextWindow component is included within the SuperWindow component.

CONTEXTWINDOW

The ContextWindow works exactly the same as mx.core.Window in that you simply create an instance, set properties, and call the open() method.

```
private function createContextWindow():void{
    var w:ContextWindow = new ContextWindow();
    w.width=300;
    w.height=200;
    w.open();
}
```

CONTEXTWINDOW

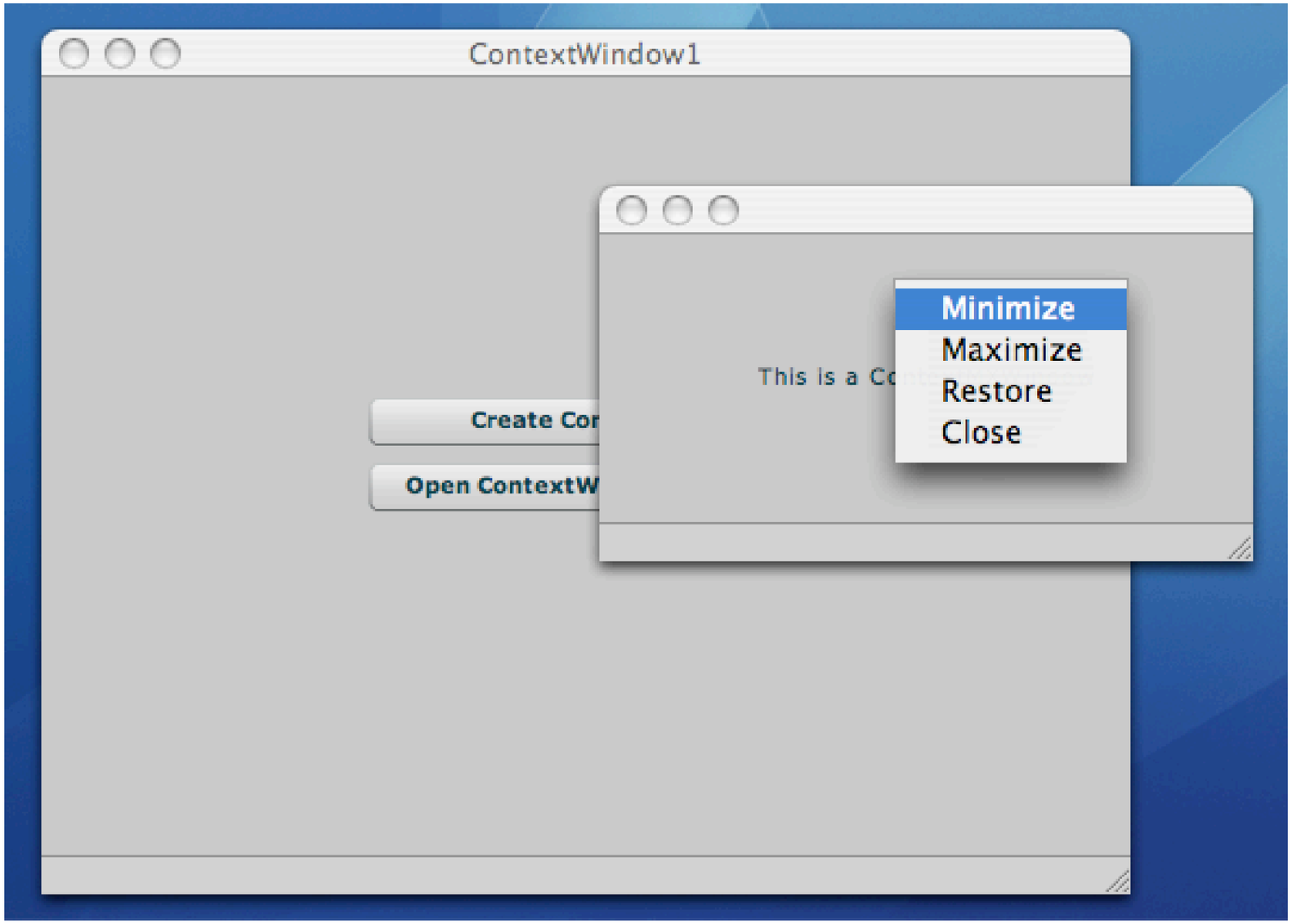
You could also create an MXML component from the ContextWindow

```
<?xml version="1.0" encoding="utf-8"?>
<ContextWindow
xmlns="com.everythingflex.air.components.*"
      layout="absolute"
      xmlns:mx="http://www.adobe.com/2006/mxml"
      width="300" height="150">

      <mx:Label text="This is a ContextWindow"
            horizontalCenter="0"
verticalCenter="0" />

</ContextWindow>
```

CONTEXTWINDOW



ContextWindow1

ALERTWINDOW

The AlertWindow component was created to add cross platform “toast” style alerts to AIR applications.

The functionality of the AlertWindow component is included within the SuperWindow component.

ALERTWINDOW

In additional to all of the properties of mx.core.Window, AlertWindow has two optional properties.

- delayTime:int = 3;
  - the time in which the window will remain on screen before retreating
- notify:Boolean = true;
  - will cause the AlertWindow to bounce only when delayTime=0

ALERTWINDOW

Create with ActionScript.

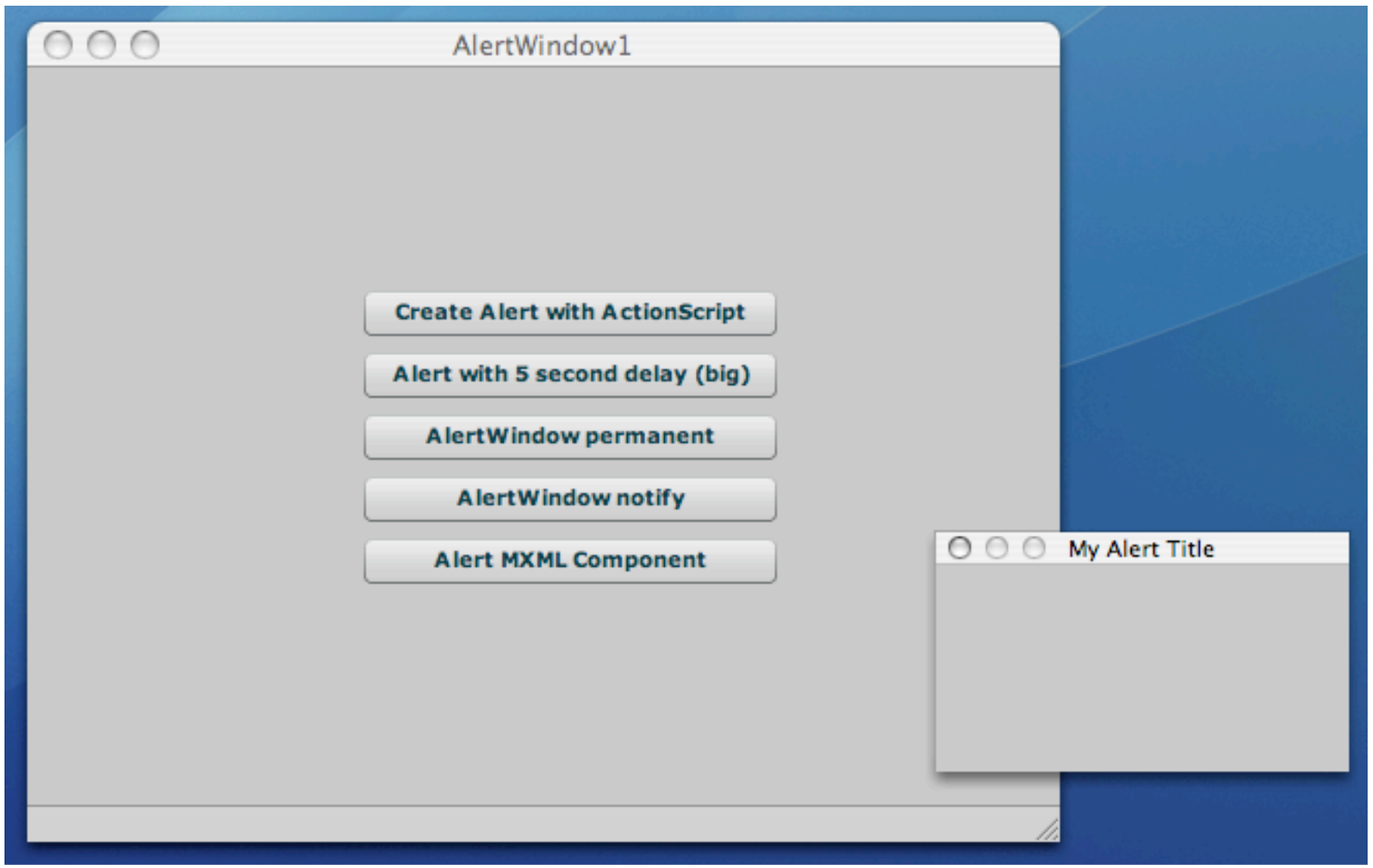
```
private function createAlertWithAS():void{
var a:AlertWindow = new AlertWindow();
     a.title = "My Alert Title";
     a.height = 100;
     a.width = 200;
     a.open();
}
```

ALERTWINDOW

You could also create an MXML component from the AlertWindow

```
<?xml version="1.0" encoding="utf-8"?>
<AlertWindow
xmlns="com.everythingflex.air.components.*"
xmlns:mx="http://www.adobe.com/2006/mxml"
layout="absolute"
title="My Window" delayTime="5" width="300"
height="100">
    <mx:Button label="Button" x="116.5" y="10"/>
    <mx:ColorPicker x="10" y="10"/>
    <mx:ComboBox x="69" y="40"/>
</AlertWindow>
```

ALERTWINDOW



AlertWindow I

CONNECTIONMANAGER

If your application requires an Internet connection, the ConnectionManager class is the easiest way to create a connection test.

CONNECTIONMANAGER

In ConnectionManager's constructor can accept 3 optional arguments.

- showMessage:Boolean = true;
  - will show an Alert window when a connection failure occurs
- connectionURL:String = "http://www.google.com";
  - the URL that s attempted
- message:String = "This application requires .....";
  - the message that is shown to the user in the Alert

CONNECTIONMANAGER

Basic

```
private var cm:ConnectionManager = new ConnectionManager();
```

Custom Settings

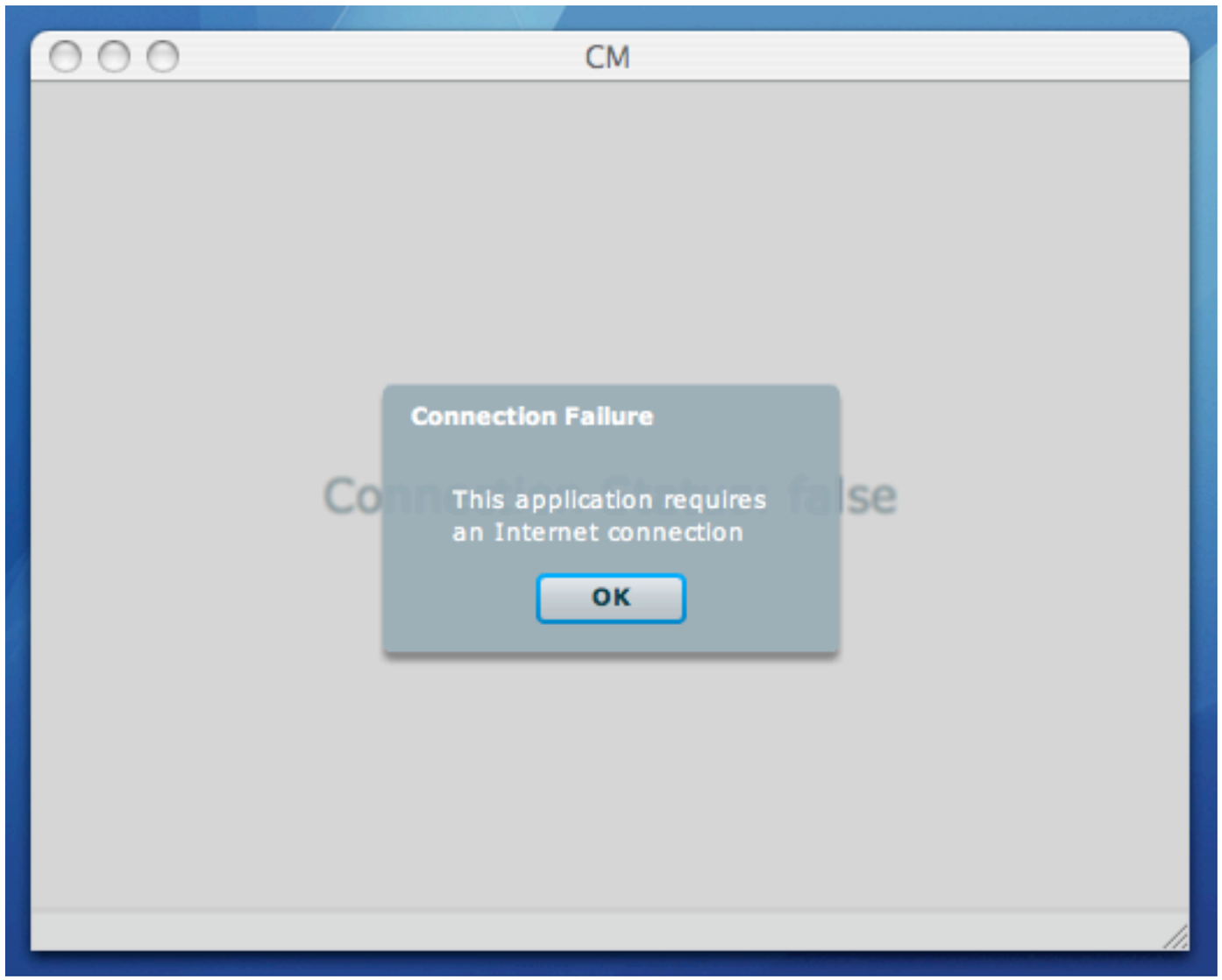
```
private var cm:ConnectionManager = new ConnectionManager(true,
"http://adobe.com","Get Online!");
```

CONNECTIONMANAGER

Properties

- connectionURL:String
  - the connectionURL currently assigned to the ConnectionManager
- isConnected:Boolean
  - current connection status

CONNECTIONMANAGER



CM

UPDATEMANAGER

I built this very simple UpdateManager to make it easy to keep your applications up up to date once they have been distributed to your clients.

UPDATEMANAGER

In UpdateManager's constructor requires one argument and has an additional optional argument.

- versionURL:String;
  - URL to the version.xml file
- autoCheck:Boolean = true;
  - if true, will automatically check for update on instantiation of the UpdateManager

UPDATEMANAGER

Basic

```
private var cm:ConnectionManager = new ConnectionManager();
```

Custom Settings

```
private var cm:ConnectionManager = new ConnectionManager(true,
"http://adobe.com","Get Online!");
```

UPDATEMANAGER

Properties

- currentVersion:String
  - the current version of the application

UPDATEMANAGER

version.xml properties

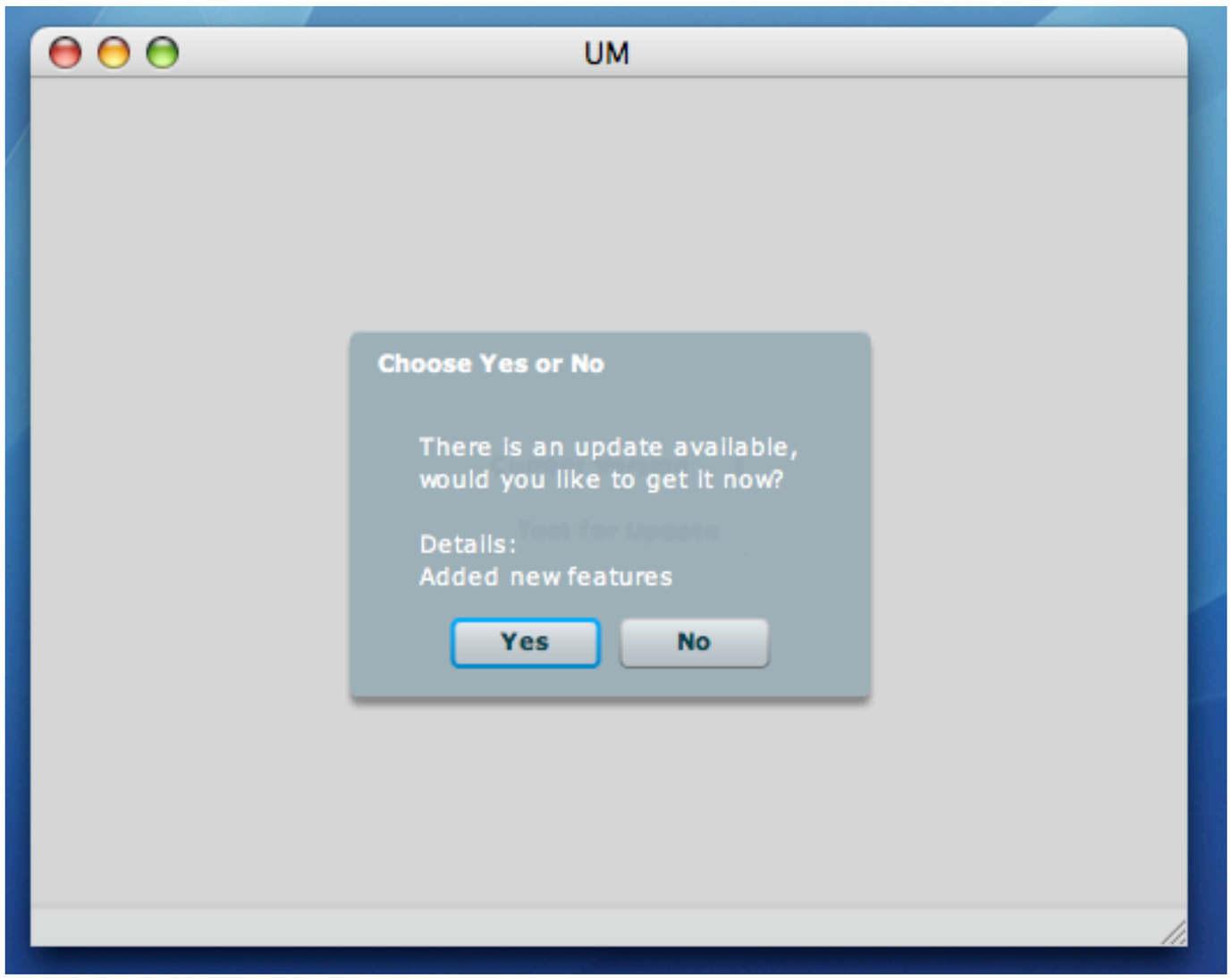
- currentVersion: the version of the new AIR file
- downloadLocation: URL to new AIR file
- forceUpdate: if true application will update itself without user interaction
- message: shows in the update Alert (only if forceUpdate is false

UPDATEMANAGER

version.xml sample

```
<?xml version="1.0" encoding="ISO-8859-1"?><currentVersion
version="1.1"
  downloadLocation="http://www.everythingflex.com/AIR/UM/UM.air"
  forceUpdate="false"
  message="Added new features"/>
```

UPDATEMANAGER



UM

EVERYTHINGFLEXAIR1.SWC

100% FREE!

http://blog.everythingflex.com/apollo/components-air

QUESTIONS & CONTACT INFO