

Introduction to Adobe Flex 3

Miscellaneous Topics

Trademarks

ChikaraDev and the ChikaraDev logo are trademarks of ChikaraDev. Such trademarks may be registered in the United States or in other jurisdictions, including internationally. This manual may include trademarks, service marks, or trade names of Adobe Systems Incorporated, Inc. and other companies. Such trademarks, service marks, or trade names may be registered in the United States or in other jurisdictions, including internationally.

Third-Party Information

This manual contains information such as links to third-party websites that are not under the control of ChikaraDev, and ChikaraDev is not responsible for the content on any linked site. If you access a third-party website mentioned in this manual, then you do so at your own risk. ChikaraDev has provided these links only as a convenience, and the inclusion of the link does not imply that ChikaraDev endorses or accepts any responsibility for the content on those third-party sites.

Copyright © 2008 ChikaraDev. All rights reserved.

The software described in this manual is provided under an agreement with Adobe Systems Incorporated, and such software can only be used in accordance with the terms of the agreement provided by Adobe Systems. Software code described and provided in this manual is provided under an agreement with ChikaraDev. The software can only be used in accordance with the terms of the agreement.

No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, photocopying, manual, optical, recording, or otherwise, outside the license agreement accompanying these materials, without the prior written permission of ChikaraDev. ChikaraDev claims copyright in this program and documentation as an unpublished work, revisions of which were first licensed on the date indicated in the foregoing notice. Claim of copyright does not imply waiver of other rights of ChikaraDev and its subsidiaries.

Information in this manual may change without notice and does not represent a commitment on the part of ChikaraDev.

NOTICE OF LIABILITY

This information in these training materials is distributed on an “AS IS” basis, without warranty of any kind, either express or implied. While every precaution has been taken in the preparation of these materials, neither ChikaraDev nor its licensors shall have any liability to any person or entity with respect to liability, loss, or damage caused or alleged to be caused directly or indirectly by the instructions contained in these materials or by the computer software and hardware products described herein.

First Edition: July 2008 - ChikaraDev - Cupertino, CA 95014 USA

Alert “Parent” Parameter

When you popup an message box using the `mx.controls.Alert.show()` method, the method has this signature:

```
public static function show(text:String = "", title:String = "", flags:uint = 0x4, parent:Sprite = null, closeHandler:Function = null, iconClass:Class = null, defaultButtonFlag:uint = 0x4):Alert
```

The fourth parameter is the “parent” of the Alert box. It is the object upon which the Alert control centers itself.

Sample application **AlertParentParam.mxml** shows displaying an Alert centered over the application, and also centered over three VBox controls using this fourth “parent” parameter.

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" width="100%" height="100%">
  <mx:Script>
    <![CDATA[
      import mx.controls.Alert;
    ]]>
  </mx:Script>
  <mx:HBox>
    <mx:Button label="Application"
      click="mx.controls.Alert.show('App', 'App', Alert.OK, this);"/>
    <mx:Button label="Red"
      click="mx.controls.Alert.show('Red', 'Red', Alert.OK, redVB);"/>
    <mx:Button label="Green"
      click="mx.controls.Alert.show('Green', 'Green', Alert.OK, greenVB);"/>
    <mx:Button label="Blue"
      click="mx.controls.Alert.show('Blue', 'Blue', Alert.OK, blueVB);"/>
  </mx:HBox>
  <mx:VBox id="redVB" width="500" height="200" backgroundColor="0xFF0000"/>
  <mx:VBox id="greenVB" width="500" height="200" backgroundColor="0x00FF00"/>
  <mx:VBox id="blueVB" width="500" height="200" backgroundColor="0x0000FF"/>
</mx:Application>
```

e4x Syntax Accessing XML Data Node Levels

When using e4x syntax to bring in data from XML, XMLList, and XMLListCollection objects, if the data being accessed is at the first “node level” you can use a single “dot”, but if the data is at more than one node level, you need two dots. You DO NOT use additional dots for additional node levels, such as three dots for the third level.

Sample application **XMLNumNodes.mxml** illustrates this:

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" width="100%" height="100%">
  <mx:Script>
    <![CDATA[
      include "data/actionScriptData/numnodes1.as";

      import mx.collections.XMLListCollection;

      [Bindable] private var level1XLC:XMLListCollection =
        new XMLListCollection(oneLevel.subOne);

      [Bindable] private var level1TwoDotXLC:XMLListCollection =
        new XMLListCollection(oneLevel..subOne);

      [Bindable] private var level2XLC:XMLListCollection =
        new XMLListCollection(twoLevel..subTwo);

      [Bindable] private var level3XLC:XMLListCollection =
        new XMLListCollection(threeLevel..subThree);

      [Bindable] private var level3OneDotXLC:XMLListCollection =
        new XMLListCollection(threeLevel.subThree);
    ]]>
  </mx:Script>
  <mx:ComboBox dataProvider="{level1XLC}"/>
  <mx:ComboBox dataProvider="{level1TwoDotXLC}"/>
  <mx:ComboBox dataProvider="{level2XLC}"/>
  <mx:ComboBox dataProvider="{level3XLC}"/>
  <mx:ComboBox dataProvider="{level3OneDotXLC}"/>
</mx:Application>
```

Increment ++ and Decrement – Pre/Post Position

In this example, before age1 is incremented, its old value is assigned to age1POST. After age2 is incremented, its new value is assigned to age2PRE.

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      [Bindable] private var age1:uint = 30;
      [Bindable] private var age2:uint = 40;
      [Bindable] private var age1POST:uint = age1++;
      [Bindable] private var age2PRE:uint = ++age2;
    ]]>
  </mx:Script>
  <mx:Label text="age1 (incremented): {age1}"/>
  <mx:Label text="age2 (incremented): {age2}"/>
  <mx:Label text="age1 Post increment: {age1POST}"/>
  <mx:Label text="age2 Pre increment: {age2PRE}"/>
  <mx:Text text="Notice how age1 is incremented AFTER the value of age1POST is set, so age1POST has the old value of age1"/>
</mx:Application>
```

Using Boolean in switch Conditional Constructs

This example shows how to use **switch** to enable you to use 1, true, or yes as Boolean. This example also illustrates “fall-through” in **switch** conditionals, meaning **return true;** will be executed if str equals 1, true, or yes, because we used no **break**.

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      static public function parseBoolean(str:String):Boolean {
        switch(str) {
          case "1":
          case "true":
          case "yes":
            return true;
            break;
          case "0":
          case "false":
          case "no":
            return false;
            break;
          default:
            return Boolean(str);
        }
      }
    ]]>
  </mx:Script>
  <mx:Label id="myLabel"/>
  <mx:Button label="1 is true here" click="myLabel.text=parseBoolean(String(1))?it is true:'it is false;"/>
  <mx:Button label="true is true here" click="myLabel.text=parseBoolean(String('true'))?it is true:'it is false;"/>
  <mx:Button label="yes is true here" click="myLabel.text=parseBoolean(String('yes'))?it is true:'it is false;"/>
  <mx:Button label="0 is true here" click="myLabel.text=parseBoolean(String(0))?it is true:'it is false;"/>
  <mx:Button label="false is true here" click="myLabel.text=parseBoolean(String('false'))?it is true:'it is false;"/>
  <mx:Button label="no is true here" click="myLabel.text=parseBoolean(String('no'))?it is true:'it is false;"/>
</mx:Application>
```

This example shows how to use the switch conditional with equality operators:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  creationComplete="init();">
  <mx:Script>
    <![CDATA[
      private function init():void{
        trace(examineString(100, 200));
        trace(examineString(200, 100));
        trace(examineString(100, 100));
      }

      static public function examineString(arg1:uint, arg2:uint):String {
        switch(arg1<arg2?0:arg1>arg2?1:-1) {
          case (0):
            return "less than";
            break;
          case (1):
            return "greater than";
            break;
          default:
            return "equals";
        }
      }
    ]]>
  </mx:Script>
</mx:Application>
```

Associative Array Key Ordering

This example shows how keys in associative arrays are randomly ordered. Refresh app to see the order change.

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  creationComplete="init();">
  <mx:Script>
    <![CDATA[
      [Bindable] private var obj:Object = {
        one: 1, two: 2, three: 3, four: 4, five: 5, six: 6, seven: 7, eight: 8, nine: 9, ten: 10
      };

      private function init():void{
        for(var key:String in obj){
          txt.text += "key: " + key + " *** val: " + obj[key] + "\n";
        }
      }
    ]]>
  </mx:Script>
  <mx:TextArea id="txt" width="200" height="200"/>
</mx:Application>
```