

Introduction to Adobe Flex 3

Guide to Getters and Setters

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 ChikaraDev claims copyright in this program and documentation as an unpublished work, revisions of which were first licensed on the date indicated in the foregoing notice.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

Introduction to Getters and Setters in Adobe Flex

Data hiding is a key concept in the world of object oriented design, because usually some data you want users to be able to access more freely, and some data you don't want users to access freely, or at all.

Some data is used internally by the application, and changing that data in inappropriate ways could cause the application to operate improperly. Other times you may want to validate data before it is changed, check data before it is returned to the user.

Data hiding is accomplished in object oriented programming is by making data "private" wherever possible, and allowing users to retrieve the data or to change the data only through "accessor methods". If accessor methods were not used, and you relied solely on your use of access modifiers such as private, protected and public, what if you used public for a variable that really should be private?

In Adobe Flex 3, accessor methods are implemented as special functions called "getters" and "setters". Getters are used to "get" the data, and setters are used to "set" the data. The actual implementation of getters and setters is somewhat unique because their use allows you to manipulate data via "variables", when in the background those variables may not directly exist! Let's look at an example.

First we'll create a custom component based on ComboBox with three private variables. Often in Flex you start private variable names for which a set and get function is defined with an underscore, just to identify them as private. This is not necessary but is widely practiced.

```
<?xml version="1.0"?>
<mx:ComboBox xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      private var stateArrayShort:Array = ["AK", "AL"];
      private var stateArrayLong:Array = ["Arkansas", "Alaska"];
      private var _shortNames:Boolean = true;

      public function set shortNames(val:Boolean):void {
        _shortNames = val;
        if (_shortNames) {
          this.dataProvider=stateArrayShort;
        }
        else {
          this.dataProvider=stateArrayLong;
        }
      }

      public function get shortNames():Boolean{
        return _shortNames;
      }
    ]]>
  </mx:Script>
</mx:ComboBox>
```

Next we will create an application to access one of the private variables using its getter and setter.

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns:comp="*">
  <mx:Script>
    <![CDATA[
      private function clickHandler():void{
        cbx1.shortNames = !cbx1.shortNames;
        cbx2.shortNames = !cbx2.shortNames;
      }
    ]]>
  </mx:Script>
  <comp:CustomComboBox id="cbx1" shortNames="true"/>
  <comp:CustomComboBox id="cbx2" shortNames="false"/>
  <mx:Button label="Toggle ComboBoxes" click="clickHandler()"/>
</mx:Application>
```

Examining the code we see that the setter and getter functions in the custom component have the same name as the tag property in MXML, shortNames. So when in MXML we code shortNames="true", we are actually calling the setter method named shortNames. In fact, the custom component does not even have a variable named "shortNames". There is one named "_shortNames" (starts with underscore) and it is that private variable being retrieved or changed via the get and set functions.

The name of the set and get functions does not even need to be similar to the actual private variable being retrieved and changed. Consider the following:

```
private var _allowMultipleValues:Boolean = true;

public function set moreThanOne(val:Boolean):void {
  _allowMultipleValues = val;
}

public function get moreThanOne ():Boolean{
  return _allowMultipleValues;
}
```

The set and get functions would be executed using the "moreThanOne" property in MXML:

```
<comp:myCompONE moreThanOne="true" />
<comp:myCompTWO moreThanOne="false" />
```

Using a set Function as a Quasi Constructor in MXML

You can create custom components in Flex in MXML or in ActionScript, and when using ActionScript you can define a constructor method to perform object initialization. But there is no concept of a constructor in MXML, so you cannot perform constructor initialization as in ActionScript. Using the following technique you can come close to simulating a constructor in MXML using a setter function.

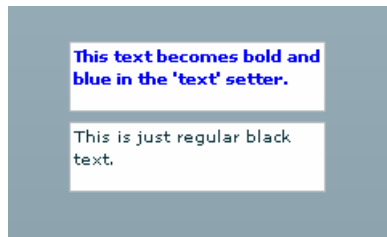
Here is the definition for a custom MXML component MyHTMLText.mxml based on TextArea:

```
<?xml version="1.0"?>
<mx:TextArea xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      override public function set text(val:String):void{
        this.htmlText = "<font color='#0000FF'><b>" + val + "</b></font>";
      }
    ]]>
  </mx:Script>
</mx:TextArea>
```

Here is a Flex application that makes use of this component:

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:comp="*">
  <comp:HTMLTextArea text="This text becomes bold and blue in the 'text' setter."/>
  <mx:TextArea text="This is just regular black text."/>
</mx:Application>
```

Notice how the setter for the “text” property has been overridden in the custom component to set the “htmlText” property whenever the “text” property is set. Here is the running application:



Even more powerful is the ability to create setters for new properties. In the following example we create a new property "values" for our custom component, and in the setter we access that new property.

HTMLTextAreaValues.mxml

```
<?xml version="1.0"?>
<mx:TextArea xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      public function set values(arr:Array):void{
        for(var a:uint=0;a<arr.length;a++){
          if(a<arr.length-1){
            this.text += arr[a] + " ";
          }else{
            this.text += arr[a] + ".";
          }
        }
      }
    ]]>
  </mx:Script>
</mx:TextArea>
```

MyHTMLTextValues.mxml

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:comp="*">
  <comp:HTMLTextAreaValues width="200"
    values="['These', 'are', 'words', 'for', 'a', 'sentence']"/>
</mx:Application>
```

