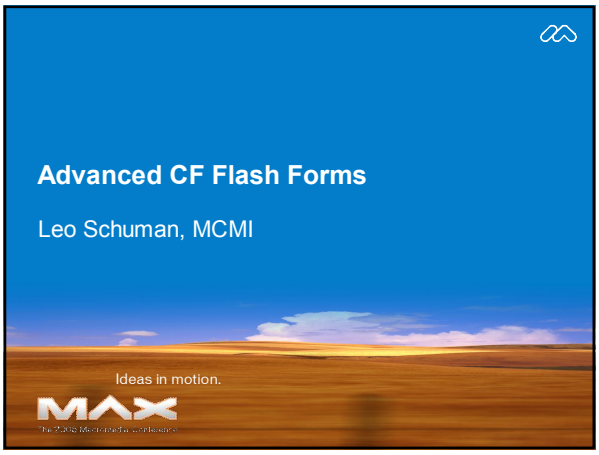

Biography

Leo Schuman
Schooner Technical Media, Inc.
www.schoonertech.com
> Macromedia Certified Master Instructor: ColdFusion, Flash, Flex
> Macromedia Certified Advanced CFMX Developer
> Macromedia Certified Flash MX 2004 Developer
> Sun Certified Java Programmer

Starting with ColdFusion 3.1, Leo has worked as a ColdFusion, Flash, and Flex instructor, developer, courseware writer, and frustrated lounge act for the past eight years.

Introduction

- Like Javascript with HTML and CSS ("DHTML"), CF Flash Forms support an event-driven programming model … but without all the cross-browser compatibility nightmares that will *never* go away in a competitive, multi-browser market.
- Developers can now create rich form interactivity, once, and display it across the internet
  - the Flash Player runtime is installed in 97.6% of web browsers worldwide.
  - http://www.macromedia.com/software/player_census/flashplayer/

Flash Forms and the Flash Component API

- How are CF Flash Forms different than HTML?
  - 100% cross-browser compatible client-side scripting
  - Compiled binary (SWF) in Flash Player runtime
- Common Flash Component properties

enabled	**visible**	**text**
value	**selectedItem**	**editable**

- Common Flash Component methods
  setFocus

ActionScript in a Nutshell: Overview

- ActionScript and Javascript are both EMCAScript
  - http://en.wikipedia.org/wiki/ECMAScript
- CF Flash Forms use ActionScript 2.0
  - case sensitive
  - some keyword restrictions (new, createobject, etc.)
  - function definitions now supported (*updater ! ! !*)
  - CF Struct equivalent is an AS2 Object
  - AS2 Arrays begin at 0

ActionScript in a Nutshell: Code Sample 1 of 3

```
var thisClient:Object = {}; // 'new' and createobject restricted
thisClient.clientName = tinClientName.text;
thisClient["creditCheck"] = cbxCreditStatus.checked;
thisClient.creditLimit = tinCreditLimit["text"];
var aClients:Array = [];
aClients.push(thisClient);
for(var i:Number = 0; i < aClients.length; i++) {
  for(var prop:String in aClients[i])       {
      // display each property value in a text area
      someCFTextArea.text += aClients[i][prop] + "/n";
  }
}
```

ActionScript in a Nutshell: Code Sample 2 of 3

```
function checkClientCredit(client:Object):String  {
  if(client.creditCheck && client.creditLimit >= 5000) {
       return "Credit Confirmed.";
  }
  else if( !(client.creditCheck) || client.creditLimit <= 0) {
       return "Credit Denied.";
  }
  else {
       return "Call to Confirm.";
  }
}
tarDisplayArea.text = checkClientCredit(aClients[0]);
```

ActionScript in a Nutshell: Code Sample 3 of 3

```
function showSelectedCheckBox():Void {
  var checkBoxCount:Number = 3;
  for(var i:Number = 1; i <= checkBoxCount; i++) {
       var thisName = "check" + i;
       if(i == itemSelect.value) {
            _root[thisName].enabled = true;
       } else {
            _root[thisName]["enabled"] = false;
       }
  }
}
```

Event Handlers: What and Where?

- What's an "event"?
  - a signal in the Flash Player stating either the user or the system did something
- What's an "event handler"?
  - code statements and/or functions executed when a specific event is broadcast by a specified object
- Where do these statements or function calls go?
  - as the values assigned to event handler attributes in CF Flash Form tags

```
<cfinput type="text" name="in1"
   onblur="out1.text=in1.text.toUpperCase();">
<cfselect name="itemSelect"
   onchange="showSelectedCheckBox();">
```

CF Flash Form Event Handler Summary: 1 of 3

- **<cfform>**
  - **onError**: If the cfinput form element fails to validate
  - **onSubmit**: If the cfform tag is submitted
  - **onLoad**: (*updater*) When the form initially loads
- **<cfformgroup>**
  - **onChange**: If the TabNavigator group tab or accordion leaf changes
- **<cfinput>**
  - **onKeyUp**: The key released while control has focus
  - **onKeyDown**: The key pressed while control has focus
  - **onMouseUp**: The mouse button pressed while control has focus
  - **onMouseDown**: The mouse button released while control has focus
  - **onChange**: User edits characters (datefield, password, text)
  - **onClick**: User clicks (button, checkbox, image, radio, reset, submit)
  - **onFocus**: (*updater*) When a user tabs to or clicks in the component
  - **onBlur**: (*updater*) When a user tabs or clicks away from the component

CF Flash Form Event Handler Summary: 2 of 3

- **<cftextarea>**
  - **onKeyUp**: The key released while control has focus
  - **onKeyDown**: The key pressed while control has focus
  - **onMouseUp**: The mouse button pressed while control has focus
  - **onMouseDown**: The mouse button released while control has focus
  - **onChange**: When a user adds/removes characters
  - **onFocus**: (*updater*) When a user tabs to or clicks in the component
  - **onBlur**: (*updater*) When a user tabs or clicks away from the component
- **<cfselect>**
  - **onKeyUp**: The key released while control has focus
  - **onKeyDown**: The key pressed while control has focus
  - **onMouseUp**: The mouse button pressed while control has focus
  - **onMouseDown**: The mouse button released while control has focus
  - **onChange**: When a user selects a new value in the control
  - **onClick**: When a user clicks on the control
  - **onFocus**: (*updater*) When a user tabs to or clicks in the component
  - **onBlur**: (*updater*) When a user tabs or clicks away from the component

CF Flash Form Event Handler Summary: 3 of 3

- **<cfgrid>**
  - **onChange**: When a user selects a grid row
  - **onFocus**: (*updater*) When a user tabs to or clicks in the component
  - **onBlur**: (*updater*) When a user tabs or clicks away from the component
- **<cfcalendar>**
  - **onChange**: When a user selects a date on the calendar
  - **onFocus**: (*updater*) When a user tabs to or clicks in the component
  - **onBlur**: (*updater*) When a user tabs or clicks away from the component

Adding ActionScript to CF Flash Forms

- inline statements

```
<cfinput type="button" value="bye!"
   onclick="this.enabled=false;" />
```

- code injection

```
<cfsavecontent name="vanish">
   this.enabled=false;
</cfsavecontent>
<input type="button" value="bye!" onclick="#vanish#" />
```

- function calls (*updater*)

```
<cfformitem type="script">
   function hideButton():Void  {
     btnBye.enabled = false;
   }
</cfformitem>
<input type="button" name="btnBye" value="bye!"
   onclick="hideButton();" />
```

Exercise 1 : Writing and Calling Form Functions

- Key Points
  - function definitions now allowed in CF Flash Forms
  - "code injection" still works, but no longer necessary

Exercise 2 : Combining Standard and Non-Standard Form Validation

- Key Points
  - <cfform> now supports an **onload** event, providing a central place to initialize the state of your form
  - CF variables can be written into AS2 script blocks, but there may be a performance hit due to recompile

Exercise 3 : Dynamically referring to CF Flash Form Components

- Key Points
  - **_root** is an absolute reference to the SWF (form) itself
  - CF Flash Form component names are **properties** of **_root**
  - AS2 supports *dynamic property references*, just like CF Structs
- A ColdFusion Analogy

```
<cfset stUser = structNew()>
<cfset stUser.userID = "ABC123">
<cfset propertyName = "userID">
<cfoutput>
   #stUser[propertyName]#
</cfoutput>
```

displays → ABC123

Resources

- **http://www.macromedia.com/devnet**
  - Advanced CF Flash Form Techniques
  - Creating Better Forms Faster with ColdFusion MX 7
- **http://www.asfusion.com/blog/category/cfform**
  - Flash Remoting from CF Flash Forms
- **http://livedocs.macromedia.com**
  - Best Practices for CF Flash Forms