Flex in a Week, Flex 4.5
Video 2.01: Implementing event handlers

Flex is an event-triggered technology.

This means that it relies on user or system interaction to drive application behavior.

In this video you will learn about events and how to handle them inline in the MXML code as well as in ActionScript event handlers.

This Employee Portal: Vehicle Request Form is the sample application that I will use in the next few videos to illustrate the concepts and implementation for handling events.

I will show you how to capture a user click on a date in the DateChooser component and then display an alert message.

Let's start by defining events.

An event indicates that something has happened in your application.

There are framework-initiated events that are the result of code execution.

For instance, the initialize event exists for all UI components and will dispatch when the component has finished its construction but before all of the component's immediate children have been laid out.

The creationComplete event for a component dispatches after the initialize event and indicates that the component has been created, laid out, and is visible.

The component's show event dispatches every time the component goes from invisible to visible.

These are just three of the common system events that exist.

UI components also have user-initiated events that fire when a user interacts with the component.

For instance, they may click on a component, change its value or mouse over it.

Available events for a component are listed in the ActionScript 3.0 Reference.

I am selecting Help > Flash Builder Help to open the Adobe Community Help window.

I am searching for the Spark Button component and then expanding the Search Options to select the ActionScript 3.0 Reference.

When I select the Button component you can see the class information.

I'm expanding the viewing area and clicking on the Events link to reveal all of the available events for the Button control.

Note that the event names closely match their functionality.

Note that Flex events do not strictly adhere to the standards defined in the W3C events specification, but they are sufficiently similar to ensure a consistent implementation.

You can look at this web site for more information about the DOM Level 3 events specification.

This is a demo application that illustrates system and user events in action.

I will review the code in a minute, but first I want to run the application.

You can see that without any interaction from me, the user, the application is already indicating that some system events have run.

When I mouse over and out of the Button control, you can see that these events are also handled in the application display.

Interestingly, the Button control is preinitialized, initialized and created before the Application container.

Containers create and lay out their children before they create themselves.

Back in the code, you can see that I have placed one event on the Application container and multiple events on the Button component.

Each of the events is handled by a function named addToTextArea(), which is defined here.

The function takes a string value and displays it in the TextArea control named reportEvents.

The demonstration showed you an example of handling an event with a function.

I will get back to that in a bit.

First let me discuss generally how you can implement event handlers.

Event handlers are actions and behaviors you want to be executed in response to an event.

You will commonly also hear the term event listener used interchangeably with event handler, although they are not exactly the same.

You will learn more about that in another video.

There are two ways to implement event handlers.

First, you can place the ActionScript code inline in the MXML tag.

Alternatively, you can create a function in a Script block.

I will show you both options.

This is the starter code for the Employee Portal: Vehicle Request Form.

I am switching to Design mode and selecting the first DateChooser control, which represents the pickup date.

In the Properties view, I am assigning an id property for the DateChooser instance to a value of pickupDate.

You will need to give class instances an id property value if you want to control them in ActionScript.

Notice that the variable name does start with a lowercase p to indicate that it is an instance.

This is a way to uniquely identify a component for use in ActionScript.

I am switching back to Source mode and then creating a Script block.

I am using the import statement to import the mx.controls.Alert class.

You will learn more about this syntax in future videos.

For now, just understand that you are importing the Flex MX Alert class so that you can use it to create a popup alert message.

Now I am locating that DateChooser instance again that is the pickupDate instance using the Outline view, which makes it easier to locate elements in your code.

You can see that the id property is on the first line of the tag, which follows the coding convention that we're using in this series.

Now, I'm going to add a change event to it.

For the event handler, I am adding inline ActionScript to create an instance of the Alert class and call its show() method.

I am double-clicking on the Editor tab so that you can see the code better.

Remember that the Alert.show() method will display a message in the Alert dialog popup.

I am adding a literal string of 'You have selected '.

I am using single quotes because double quotes are already used around this value.

Note that there is a literal space after the word “selected” and inside the single quote.

I am adding a plus sign to indicate that the following string should be concatenated.

The DateChooser class has a property named selectedDate that indicates the date that the user has selected in the control.

I am referencing the DateChooser instance that I named pickupDate and then adding a period to specifically access the selectedDate property in the Flash Builder code assist tool.

The selectedDate property is data typed to the Date class but I need the value to be a string so I am using the class toDateString() method, which will convert the date into a string.

I am saving the file and running the application.

When I click on a date in the pickupDate instance, you can see that an Alert window appears with the message I just defined.

Note that the second DateChooser instance does not yet work.

This is the code for the second DateChooser instance that represents the car return date.

I am adding an id property to it with a value of returnDate.

I am also copying the change property and value from the pickupDate instance and pasting it into the DateChooser for the returnDate.

Lastly, I am updating the instance reference in the Alert.show() method to reference the returnDate instance.

When I save the file and run the application, you can see that the Alert window shows the proper date selected for each DateChooser instance.

You can imagine that writing all of the code in the click event can easily become difficult to manage.

This is why you have two choices for integrating ActionScript in your MXML application.

Writing ActionScript within the MXML tag is known as “inline ActionScript,” and can be used in two cases.

When you reference an instance name and property in a binding, you are actually referencing the two values in ActionScript.

The code that is run for an MXML event, in my example, the change event, must only contain ActionScript commands.

The second way to integrate ActionScript into your MXML application is to add it in an MXML Script block.

By convention, any event handler with more than one line of code should be referenced as an event handler in a Script tag block.

This keeps code cleaner and enables the code to be reused.

Back in the car request example, you can see that the code in the change events for both DateChooser instances only differ by the reference to the instance id – pickupDate and returnDate.

Although this is only one line of code, moving the code to a function will allow me to reuse it for both instances.

Inside of the MXML Script block, I am creating a private function named dateChangeHandler() that takes no parameters and returns a void data type.

The word “private” is an access modifier and determines that this function can only be called from references in the same class instance.

The “function” keyword declares that dateChangeHandler() is a function and the “void” return type states that this function will not return any values.

I am cutting the value in the change event of the pickupDate instance and pasting it between the curly braces of the dateChangeHandler() function.

The semicolon that I am placing at the end of the line tells the parser that it has reached the end of a statement.

I am copying the name of the function and now I am updating the event handler of the change event to be dateChangeHandler().

Keep in mind that an event handler really is just a function that is called from an event.

I am deleting the Alert.show() method from the change event of the returnDate instance and also replacing it with the dateChangeHandler() function.

When I save the file and run the application, the Alert message still works properly when I click on a pickup date.

However, when I click on the returnDate instance, you will see the value of the pickupDate control.

This is because the event handler function explicitly references the pickupDate instance not the returnDate instance.

I am refreshing the application and clicking the returnDate control without first clicking the pickupDate instance.

You can see that an error appears because the pickupDate has not yet been selected.

I will correct this in the next video.

For your next step, work through the exercise titled “Handling a user event”.