

# Creating Custom Events and Dispatching Data

## Create a user event and handler

1. Open **Options.mxml** .
2. Locate the first **CheckBox** and add a `click` event having an event listener of `clickHandler`. Pass `event` as a parameter to the method.

```
<mx:CheckBox x="132" y="10"  
    label="Dance Floor"  
    click="clickHandler(event)"/>
```

3. Add a `click` event having a value of `clickHandler(event)` to each of the **CheckBox** controls.
4. After the beginning **Canvas** tag, add a **Script** block.
5. Within the **Script** block create a private function named `clickHandler` that takes one parameter named `event` datatyped as `Event`. The function returns `void`.

```
private function clickHandler(event:Event):void{  
  
}
```

6. Within the function, assign the `event.target.label` to a local variable named `selection` datatyped as `String`.

```
var selection:String = event.target.label;
```

## Create a component event

7. After the **Script** block, create **Metadata** tags.

```
<mx:Metadata> </mx:Metadata>
```

8. Between the **Metadata** tags add an Event named `optionSelected` of type `flash.events.Event` using bracket notation .

```
<mx:Metadata>
    [Event(name="optionSelected",
type="flash.events.Event")]
</mx:Metadata>
```

## Dispatch an event

9. Locate the **clickHandler** method within the **Script** block.
10. After the **selection** variable, create a variable named `optionSelectEvent` datatyped as `Event`. Assign a new `Event` with a parameter named `optionSelected` to the `optionSelectEvent` variable.
11. Use the `dispatchEvent()` method to dispatch the `optionSelectEvent`.

```
var optionSelectEvent:Event = new Event("optionSelected");
dispatchEvent(optionSelectEvent);
```

12. Save the file.

## Handle the component event in the main application

13. Return to **AdobeODT.mxml** and locate the **Options** component.
14. Add an `optionSelected` event having an event handler named `optionHandler`. Pass `event` as a parameter to the method.

```
<comp:Options x="0" y="26"
    optionSelected="optionHandler(event)"/>
```

15. Locate the **Script** block.
16. Before the end of the **Script** block, create a private function named `optionHandler` that takes one parameter named `event` datatyped as `Event`. The function returns `void`.

```
private function optionHandler(event:Event):void{
}
```

17. Within the function invoke the `show()` method of the `Alert` class. Pass `"A checkbox was checked"` as the parameter.

```
Alert.show("A checkbox was checked");
```

18. Save the file and run.

When you click a checkbox you should see an alert message.

## Create a custom event class

19. In Flex Builder **Navigator** view, right click on the **src** folder and select **New > Folder**.

20. The **Folder name** is events.

21. Click **Finish**.

22. Right click on the **events** folder and select **New > ActionScript Class**.

23. The **Package** is events.

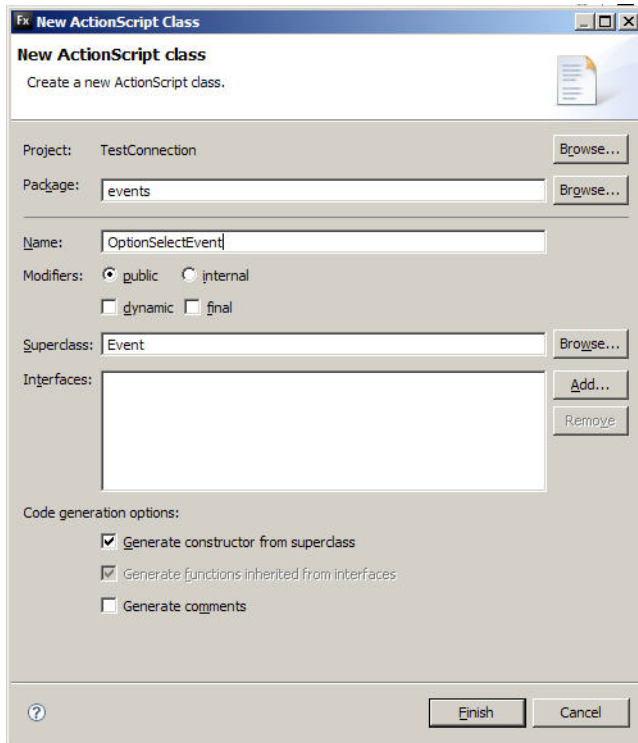
24. The **Name** is OptionSelectEvent.

25. The **Superclass** is Event

26. There are no Interfaces

27. Check **Generate constructor from superclass**.

28. Click **Finish**.



. . .

29. Create another public variable named `option` datatyped as `String`.

```
public var option:String;
```

30. Change the constructor to take two parameters:

- a. type datatyped as `String`
- b. `option` datatyped as `String`

31. Within the constructor, change the **super()** method to accept one parameter named `type`.

32. Assign the `option` parameter to the local `option` variable using the keyword `this`.

```
this.option = option;
```

Your constructor will appear like this:

```
public function OptionSelectEvent(type:String, option:String
)
{
    super(type);
}
```

```
        this.option = option;
    }
```

33. After the constructor but within the class definition, override the `public clone` function. The function takes no parameters and returns `Event`.

```
override public function clone():Event{
}
```

34. Within the function return a new `OptionSelectEvent` that has `type` and `option` as parameters.
- ```
return new OptionSelectEvent(type, option);
```

Your code should appear like this:

```
override public function clone():Event{
    return new OptionSelectEvent(type, option);
}
```

35. Save the file.

## Use the custom event class

36. Open **Options.mxml** and locate the **Script** block.
37. Before the function, import the `OptionSelectEvent` class.

```
import events.OptionSelectEvent;
```

38. Locate **Metadata** code after the **Script** block.
39. Change the **type** of the event to `events.OptionSelectEvent`.

```
[Event(name="optionSelected",
        type="events.OptionSelectEvent")]
```

40. Locate the **clickHandler** function.
41. Change the datatype of the **optionSelectEvent** variable to `OptionSelectEvent`.
42. Change the **new Event** assigned to the **optionSelectEvent** variable to `new OptionSelectEvent` passing two parameters: `optionSelected` and `selection`.

```
var optionSelectEvent:OptionSelectEvent = new
OptionSelectEvent("optionSelected",selection);

dispatchEvent(optionSelectEvent);
```

43. Save the file.

## Handle the custom event class in the main application

44. Open **AdobeODT.mxml** and locate the **Script** block.
45. After the last import statement, import the `OptionSelectEvent` class

```
import events.OptionSelectEvent;
```

46. After the current variable, create a private variable named `selectedOptions` datatyped as `ArrayCollection`. Instantiate it by creating a new `ArrayCollection`. Make the `selectedOptions` variable `Bindable`.

```
[Bindable]
public var selectedOptions:ArrayCollection = new
ArrayCollection();
```

47. Locate the **optionHandler** function and change the parameter's datatype to `OptionSelectEvent`.

```
private function
optionHandler(event:OptionSelectEvent):void{
. . .
```

48. Remove the **Alert** code within the **optionHandler** function.

## Inspect the custom event data

49. Change to the **Flex Debugging** perspective.
50. Add a breakpoint to the line with the closing curly brace of the **optionHandler** function.
51. Save the file and debug.
52. Check a checkbox. This should return you to Flex Builder. If not, return to Flex Builder.
53. Double click on the **Variables** view to expand it.
54. Expand the **event** variable. Notice the **option** variable of the custom event has the value of the checkbox you selected.

55. Double click on the **Variables** view to collapse it.
56. Stop the debugging session by click the red square.
57. Return to the **Flex Development** view.

## Add event data to an ArrayCollection

58. Within the **optionHandler** function, create a locale variable named `index` datatype as `int`.
59. To return the location of the data in the `selectedOptions` **ArrayCollection**
60. , use the `getItemIndex()` method and pass `event.option`. Assign the result to the `index` variable.  

```
var index:int = selectedOptions.getItemIndex(event.option);
```
61. After the variable, create a conditional to test the value of the `index`. If it is `-1` (negative one), add the `event.option` to `selectedOptions` using the `addItem()` method. Otherwise use the `index` to remove the data from `selectedOptions` using the `removeItemAt()` method.

```
if (index == -1){
    selectedOptions.addItem(event.option);
}else{
    selectedOptions.removeItemAt(index);
}
```

## Pass the data to another custom component

62. Locate the **ReservationForm** component and add the `selectedOptions` property having a value bound to the `selectedOptions` **ArrayCollection**.

```
<comp:ReservationForm x="368" y="114" width="318"
height="529" selectedOptions="{selectedOptions}"/>
```

63. Save the file.

## Listen for data change

64. Open **ReservationForm.mxml** and locate the beginning **VBox** tag.
65. Add a `creationComplete` property with a value of `init()`.
66. Within the **Script** block following the last `import` statement, import the `CollectionEvent` class.

```
import mx.events.CollectionEvent;
```

67. Before the end of the **Script** block, create a private function named `init` that takes no parameters. It returns `void`.

```
private function init():void{  
}
```

68. Within the function add an event listener to the selectedOptions **ArrayCollection** using the `addEventListener()` method. Pass two parameters: `CollectionEvent.COLLECTION_CHANGE` (uppercase) and `addToList`.

```
private function init():void{  
    selectedOptions.addEventListener(CollectionEvent.COLLECTION_CHANGE, addToList);  
}
```

## Display event data in the Form

69. Locate the **FormItem** tags containing the **Button** control in the **Form** container.
70. Before this **FormItem** tag add a `FormHeading` tag with a `label` property having a value of `Selected Options`.

```
<mx:FormHeading label="Selected Options"/>
```

71. Following the **FormHeading**, add `FormItem` tags with a `label` property having a value of `Selected Options`.

72. Between the **FormItem** tags, add a `List` control with an `id` property having a value of `thelist`.

73. Also add the following properties:

- c. `width` having a value of `100%`
- d. `height` having a value of `122`
- e. `fontSize` having a value of `9`
- f. `borderStyle` having a value of `none`
- g. `backgroundAlpha` having a value of `0`
- h. `leading` having a value of `0`

Your code should appear as follows:

```
<mx:FormItem label="Selected Options:">
  <mx>List id="thelist"
    width="100%" height="122"
    fontSize="9" borderStyle="none"
    backgroundAlpha="0" leading="0" />
</mx:FormItem>
```

## Populate the List control with data

74. Locate the **Script** block.

75. Before the end of the **Script** block, create a private function named `addToList` that takes one parameter named `event` datatyped as `CollectionEvent`. The function returns `void`.

76. Within the function, assign `selectedOptions` to the `dataProvider` of `thelist`.

Your code should appear like this:

```
private function addToList(event:CollectionEvent):void{

    thelist.dataProvider = selectedOptions;
}
```

77. Save the file.

78. Return to **AdobeODT.mxml** and run.

Click on one or more of the checkboxes. You should see the value of the checkbox displayed in the form. Uncheck one or more checkboxes. You should see that the value is removed from the form.

**XYZ Convention Center**  
111 49th Ave, Denver, CO 80634 Phone: 303-555-1212 Fax: 303-555-1234

**Room Reservation System**

Please select any additional amenities needed for your room. Our staff will contact you to finalize your reservation.

Microphone     Dance Floor     Food Station     Security     Extra Trash Recepticals  
 Projector     Special Lighting     Color Table Linens     Waitstaff     Additional Electrical Outlets


Rooms Available:

- Colorado Room
- Mile High Room
- Bronco Room
- Greeley Room
- Boulder Room
- Denver Room

**Contact Information**

Full Name:   
Address:   
City:   
State:   
Postal Code:   
Phone:

**Room Information**

Date Needed:  

**Selected Options**

Selected Options: Dance Floor