This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

**Creating Custom Client Events**

AJAX functionality in ASP.NET includes a complete multi-layered client-event model. The [Sys.Application](http://msdn.microsoft.com/en-us/library/bb310856.aspx) class provides events at the application level. The [Sys.WebForms.PageRequestManager](http://msdn.microsoft.com/en-us/library/bb311028.aspx) class provides events that pertain to parts of the page involved in partial-page rendering. Individual components, such as controls and behaviors, have their own events. For more information about these events, see [Ajax Client Life-Cycle Events](http://msdn.microsoft.com/en-us/library/bb386417.aspx).

ASP.NET also enables you to add events to the client life cycle. The [Sys.UI.DomEvent](http://msdn.microsoft.com/en-us/library/bb383775.aspx) class enables you to bind HTML Document Object Model (DOM) events to custom ASP.NET AJAX components. In addition, the [Sys.EventHandlerList](http://msdn.microsoft.com/en-us/library/bb383996.aspx) class lets you create new ASP.NET AJAX client events directly.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifAttaching to DOM Events

In many cases, an event you want to work with corresponds to an event defined in the HTML DOM. For example, a custom ASP.NET AJAX button control might use the **click** event of the HTML **<button>** element that it is attached to. To bind a DOM-based event to an ASP.NET AJAX application or custom component, you use the [addHandler](http://msdn.microsoft.com/en-us/library/bb310798.aspx) method of the **DomEvent** class, as shown in the following example:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl47_ctl00_ctl01_code');" \o "Copy Code)

Sys.UI.DomEvent.addHandler(element, 'click', this.myClickHandler);

You can also use the [$addHandler](http://msdn.microsoft.com/en-us/library/bb311019.aspx) shortcut, as shown in the following example:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl47_ctl00_ctl03_code');" \o "Copy Code)

$addHandler(element, 'click', this.myClickHandler);

The **addHandler** method takes three parameters: *element*, *eventName*, and *handler*. The *element* parameter is a reference to the DOM element that exposes the event. The *eventName* parameter is the name of the DOM event itself. The *handler* parameter is a reference to the function to call when the event is raised. For more information, see [Sys.UI.DomEvent addHandler Method](http://msdn.microsoft.com/en-us/library/bb310798.aspx).

To remove a handler for a DOM event, you call the [Sys.UI.DomEvent.removeHandler](http://msdn.microsoft.com/en-us/library/bb310935.aspx) method or the [$removeHandler](http://msdn.microsoft.com/en-us/library/bb397510.aspx) shortcut, passing the same parameters that you pass to **addHandler**.

|  |
| --- |
| **Description: NoteNote** |
| Event names passed to the **addHandler** and **removeHandler** functions should not include the "on" prefix. For example, use "click", not "onclick". |

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifAdding and Removing Custom Event Handlers

To add a new event handler for an event of a custom ASP.NET AJAX component, use the [addHandler](http://msdn.microsoft.com/en-us/library/bb384007.aspx) method of the [Sys.EventHandlerList](http://msdn.microsoft.com/en-us/library/bb383996.aspx) class. All client events and associated event handlers in the ASP.NET AJAX event model are stored in an **EventHandlerList** object, which is a specialized dictionary for that purpose. Every component, including the current **Application** object, has its own **EventHandlerList** instance. By adding an item to the **EventHandlerList** object, you add a new event and event handler to the associated component. Add events by using the following syntax:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl48_ctl00_ctl02_code');" \o "Copy Code)

this.get\_events().addHandler(event, handler);

The [events](http://msdn.microsoft.com/en-us/library/bb383770.aspx) property of the [Sys.Component](http://msdn.microsoft.com/en-us/library/bb397516.aspx) class returns the **EventHandlerList** instance for that component. The **events** property is inherited by the [Sys.UI.Control](http://msdn.microsoft.com/en-us/library/bb310848.aspx), [Sys.UI.Behavior](http://msdn.microsoft.com/en-us/library/bb311020.aspx), and **Sys.Application** classes. The *event* parameter is the name of the new or existing event to add a handler for. The *handler* parameter is a reference to the function to call when the event is raised. By setting the *event* parameter to a new value, you add a new event to the dictionary.

To remove a custom event from the dictionary, you use the [removeHandler](http://msdn.microsoft.com/en-us/library/bb310827.aspx) method of the **Sys.EventHandlerList** class, which takes the same parameters as **addHandler**.

To add handlers to events that are already defined in the Microsoft Ajax Library, use the **add\_** accessor for that event, as in the following example:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl48_ctl00_ctl08_code');" \o "Copy Code)

Sys.Application.add\_load(myLoadHandler);

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifRaising Custom Events

To raise a custom event, call the [getHandler](http://msdn.microsoft.com/en-us/library/bb383813.aspx) method of the **EventHandlerList** instance, passing the name of the event as a parameter. This method returns a function that aggregates all the functions that are handlers for that event. Call the returned function to raise the event, as in the following example:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl49_ctl00_ctl01_code');" \o "Copy Code)

var h = this.get\_events().getHandler('myCustomEvent')

if (h) h(this, Sys.EventArgs.Empty);

By convention, event handlers take two parameters: *sender* and *eventArgs*. The sender is the component that the event applies to, typically **this**. The *eventArgs* parameter refers to a [Sys.EventArgs](http://msdn.microsoft.com/en-us/library/bb383795.aspx) object. This object can contain information passed to the event, such as mouse coordinates. You can omit the *sender* and *eventArgs* parameters, as long as the signature of the function returned by **getHandler** matches that of all the associated handler functions. However, the recommended practice is to include the parameters, as in the example shown earlier.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifExample

**Description**

The following example creates a simple multiple-choice test with two sections. When all the questions in a section are answered, the section's background color changes. When the user clicks the button at the end of the test, a status message next to each question displays whether the answer is correct.

The application includes instances of two custom controls. The Question control attaches to an HTML **<select>** element, and the Section control attaches to a **<div>** element that contains one or more Question controls. The Question control exposes a select event, which is bound to the **onChange** event of the underlying **<select>** element by a **Sys.UI.DomEvent** instance. The Section control exposes a **complete** event, which is raised by a user-defined function when all the Question controls in the Section instance are answered.

**Code**

The following example shows the Default.aspx page that creates the component instances and handles the events.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl50_ctl00_ctl00_code');" \o "Copy Code)

<%@ Page Language="VB" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html >

<head id="Head1" runat="server">

<title>Custom Events Example</title>

</head>

<body>

<form id="form1" runat="server">

<asp:ScriptManager ID="ScriptManager1" runat="server" >

<Scripts>

<asp:ScriptReference Path="question.js" />

<asp:ScriptReference Path="section.js" />

</Scripts>

</asp:ScriptManager>

<script type="text/javascript">

// Add handler to init event

Sys.Application.add\_init(appInitHandler);

function appInitHandler() {

// create components

$create(Demo.Question, {correct: '3'},

{select: onAnswer},null, $get('Question1'));

$create(Demo.Question, {correct: '3'},

{select: onAnswer},null, $get('Question2'));

$create(Demo.Question, {correct: '3'},

{select: onAnswer},null, $get('Question3'));

$create(Demo.Question, {correct: '3'},

{select: onAnswer},null, $get('Question4'));

$create(Demo.Section, null,

{complete: onSectionComplete},null, $get('group1'));

$create(Demo.Section, null,

{complete: onSectionComplete},null, $get('group2'));

}

function onAnswer(question) {

// If all questions in this section answered,

// raise complete event

var section = question.get\_element().parentElement;

var questions = section.children;

$get(question.get\_id() + 'Status').innerHTML = '';

for (var i=0; i<questions.length; i++) {

if (questions[i].selectedIndex === -1) {

return;

}

}

$find(section.id).raiseComplete();

}

function onSectionComplete(section) {

// Change background color of <div>.

section.get\_element().style.backgroundColor = 'yellow';

}

function done() {

// Display correct answers where needed.

var c = Sys.Application.getComponents();

for (var i=0; i<c.length; i++) {

var type = Object.getType(c[i]).getName();

if (type !== 'Demo.Question') continue;

var element = c[i].get\_element();

var answer = element.selectedIndex;

var correct = $find(c[i].get\_id()).get\_correct();

var statusElement = c[i].get\_id() + 'Status';

if (answer !== correct) {

$get(statusElement).innerHTML = 'Incorrect. Try again.';

}

else

{

$get(statusElement).innerHTML = 'Correct.';

}

}

}

function resethandler() {

var c = Sys.Application.getComponents();

for (var i=0; i<c.length; i++) {

var type = Object.getType(c[i]).getName();

if (type === 'Demo.Question') {

var element = c[i].get\_element();

element.selectedIndex = -1;

var answer = element.selectedIndex;

var statusElement = c[i].get\_id() + 'Status';

$get(statusElement).innerHTML = '';

}

else if (type === 'Demo.Section') {

c[i].get\_element().style.backgroundColor = 'White';

}

}

}

</script>

<h3>Addition</h3><br />

<div id="Group1">

2 + 2 =

<select id="Question1">

<option>2</option>

<option>22</option>

<option>4</option>

<option>5</option>

</select><span id="Question1Status"></span><br />

2 + 3 =

<select id="Question2" >

<option>3</option>

<option>23</option>

<option>5</option>

<option>6</option>

</select><span id="Question2Status"></span><br />

</div><br /> <br />

<h3>Subtraction</h3><br />

<div id="Group2">

2 - 1 =

<select id="Question3" >

<option>2</option>

<option>0</option>

<option>1</option>

<option>-2</option>

</select><span id="Question3Status"></span><br />

2 - 2 =

<select id="Question4" >

<option>2</option>

<option>-2</option>

<option>0</option>

<option>-4</option>

</select><span id="Question4Status"></span><br />

</div><br /><br />

<input id="Submit1" type="button" value="Check Answers" onclick="done()" />

<input id="Reset1" type="button" value="Start Again" onclick="resethandler()" />

</form>

</body>

</html>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl50_ctl00_ctl01_code');" \o "Copy Code)

<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html >

<head id="Head1" runat="server">

<title>Custom Events Example</title>

</head>

<body>

<form id="form1" runat="server">

<asp:ScriptManager ID="ScriptManager1" runat="server" >

<Scripts>

<asp:ScriptReference Path="question.js" />

<asp:ScriptReference Path="section.js" />

</Scripts>

</asp:ScriptManager>

<script type="text/javascript">

// Add handler to init event

Sys.Application.add\_init(appInitHandler);

function appInitHandler() {

// create components

$create(Demo.Question, {correct: '3'},

{select: onAnswer},null, $get('Question1'));

$create(Demo.Question, {correct: '3'},

{select: onAnswer},null, $get('Question2'));

$create(Demo.Question, {correct: '3'},

{select: onAnswer},null, $get('Question3'));

$create(Demo.Question, {correct: '3'},

{select: onAnswer},null, $get('Question4'));

$create(Demo.Section, null,

{complete: onSectionComplete},null, $get('group1'));

$create(Demo.Section, null,

{complete: onSectionComplete},null, $get('group2'));

}

function onAnswer(question) {

// If all questions in this section answered,

// raise complete event

var section = question.get\_element().parentElement;

var questions = section.children;

$get(question.get\_id() + 'Status').innerHTML = '';

for (var i=0; i<questions.length; i++) {

if (questions[i].selectedIndex === -1) {

return;

}

}

$find(section.id).raiseComplete();

}

function onSectionComplete(section) {

// Change background color of <div>.

section.get\_element().style.backgroundColor = 'yellow';

}

function done() {

// Display correct answers where needed.

var c = Sys.Application.getComponents();

for (var i=0; i<c.length; i++) {

var type = Object.getType(c[i]).getName();

if (type !== 'Demo.Question') continue;

var element = c[i].get\_element();

var answer = element.selectedIndex;

var correct = $find(c[i].get\_id()).get\_correct();

var statusElement = c[i].get\_id() + 'Status';

if (answer !== correct) {

$get(statusElement).innerHTML = 'Incorrect. Try again.';

}

else

{

$get(statusElement).innerHTML = 'Correct.';

}

}

}

function resethandler() {

var c = Sys.Application.getComponents();

for (var i=0; i<c.length; i++) {

var type = Object.getType(c[i]).getName();

if (type === 'Demo.Question') {

var element = c[i].get\_element();

element.selectedIndex = -1;

var answer = element.selectedIndex;

var statusElement = c[i].get\_id() + 'Status';

$get(statusElement).innerHTML = '';

}

else if (type === 'Demo.Section') {

c[i].get\_element().style.backgroundColor = 'White';

}

}

}

</script>

<h3>Addition</h3><br />

<div id="Group1">

2 + 2 =

<select id="Question1">

<option>2</option>

<option>22</option>

<option>4</option>

<option>5</option>

</select><span id="Question1Status"></span><br />

2 + 3 =

<select id="Question2" >

<option>3</option>

<option>23</option>

<option>5</option>

<option>6</option>

</select><span id="Question2Status"></span><br />

</div><br /> <br />

<h3>Subtraction</h3><br />

<div id="Group2">

2 - 1 =

<select id="Question3" >

<option>2</option>

<option>0</option>

<option>1</option>

<option>-2</option>

</select><span id="Question3Status"></span><br />

2 - 2 =

<select id="Question4" >

<option>2</option>

<option>-2</option>

<option>0</option>

<option>-4</option>

</select><span id="Question4Status"></span><br />

</div><br /><br />

<input id="Submit1" type="button" value="Check Answers" onclick="done()" />

<input id="Reset1" type="button" value="Start Again" onclick="resethandler()" />

</form>

</body>

</html>

The following example shows the Question.js file that defines the Demo.Question control.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl50_ctl00_ctl02_code');" \o "Copy Code)

Type.registerNamespace("Demo");

// Constructor

Demo.Question = function(element) {

Demo.Question.initializeBase(this, [element]);

// Create a delegate for the select event.

this.\_selectDelegate = null;

}

Demo.Question.prototype = {

// correct property accessors

get\_correct: function() {

return this.get\_element().name - 1;

},

set\_correct: function(value) {

this.get\_element().name = value;

},

// Bind and unbind to select event.

add\_select: function(handler) {

this.get\_events().addHandler('select', handler);

},

remove\_select: function(handler) {

this.get\_events().removeHandler('select', handler);

},

// Release resources before control is disposed.

dispose: function() {

var element = this.get\_element();

if (this.\_selectDelegate) {

$clearHandlers(element);

delete this.\_selectDelegate;

}

Demo.Question.callBaseMethod(this, 'dispose');

},

initialize: function() {

var element = this.get\_element();

// Make sure no option is selected.

element.value = "";

// Attach delegate to select event.

if (this.\_selectDelegate === null) {

this.\_selectDelegate = Function.createDelegate(this, this.\_selectHandler);

}

Sys.UI.DomEvent.addHandler(element, 'change', this.\_selectDelegate);

Demo.Question.callBaseMethod(this, 'initialize');

},

\_selectHandler: function(event) {

var h = this.get\_events().getHandler('select');

if (h) h(this, Sys.EventArgs.Empty);

}

}

Demo.Question.registerClass('Demo.Question', Sys.UI.Control);

// Since this script is not loaded by System.Web.Handlers.ScriptResourceHandler

// invoke Sys.Application.notifyScriptLoaded to notify ScriptManager

// that this is the end of the script.

if (typeof(Sys) !== 'undefined') Sys.Application.notifyScriptLoaded();

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl50_ctl00_ctl03_code');" \o "Copy Code)

Type.registerNamespace("Demo");

// Constructor

Demo.Question = function(element) {

Demo.Question.initializeBase(this, [element]);

// Create a delegate for the select event.

this.\_selectDelegate = null;

}

Demo.Question.prototype = {

// correct property accessors

get\_correct: function() {

return this.get\_element().name - 1;

},

set\_correct: function(value) {

this.get\_element().name = value;

},

// Bind and unbind to select event.

add\_select: function(handler) {

this.get\_events().addHandler('select', handler);

},

remove\_select: function(handler) {

this.get\_events().removeHandler('select', handler);

},

// Release resources before control is disposed.

dispose: function() {

var element = this.get\_element();

if (this.\_selectDelegate) {

$clearHandlers(element);

delete this.\_selectDelegate;

}

Demo.Question.callBaseMethod(this, 'dispose');

},

initialize: function() {

var element = this.get\_element();

// Make sure no option is selected.

element.value = "";

// Attach delegate to select event.

if (this.\_selectDelegate === null) {

this.\_selectDelegate = Function.createDelegate(this, this.\_selectHandler);

}

Sys.UI.DomEvent.addHandler(element, 'change', this.\_selectDelegate);

Demo.Question.callBaseMethod(this, 'initialize');

},

\_selectHandler: function(event) {

var h = this.get\_events().getHandler('select');

if (h) h(this, Sys.EventArgs.Empty);

}

}

Demo.Question.registerClass('Demo.Question', Sys.UI.Control);

// Since this script is not loaded by System.Web.Handlers.ScriptResourceHandler

// invoke Sys.Application.notifyScriptLoaded to notify ScriptManager

// that this is the end of the script.

if (typeof(Sys) !== 'undefined') Sys.Application.notifyScriptLoaded();

The following example shows the Section.js file that defines the Demo.Section control.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl50_ctl00_ctl04_code');" \o "Copy Code)

Type.registerNamespace("Demo");

// Constructor

Demo.Section = function(element) {

Demo.Section.initializeBase(this, [element]);

}

Demo.Section.prototype = {

// Create add and remove accessors fot the complete event.

add\_complete: function(handler) {

this.get\_events().addHandler("complete", handler);

},

remove\_complete: function(handler) {

this.get\_events().removeHandler("complete", handler);

},

// Create a function to raise the complete event.

raiseComplete: function() {

var h = this.get\_events().getHandler('complete');

if (h) h(this);

},

// Release resources before control is disposed.

dispose: function() {

var element = this.get\_element();

$clearHandlers(element);

Demo.Section.callBaseMethod(this, 'dispose');

}

}

Demo.Section.registerClass('Demo.Section', Sys.UI.Control);

// Since this script is not loaded by System.Web.Handlers.ScriptResourceHandler

// invoke Sys.Application.notifyScriptLoaded to notify ScriptManager

// that this is the end of the script.

if (typeof(Sys) !== 'undefined') Sys.Application.notifyScriptLoaded();

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl50_ctl00_ctl05_code');" \o "Copy Code)

Type.registerNamespace("Demo");

// Constructor

Demo.Section = function(element) {

Demo.Section.initializeBase(this, [element]);

}

Demo.Section.prototype = {

// Create add and remove accessors fot the complete event.

add\_complete: function(handler) {

this.get\_events().addHandler("complete", handler);

},

remove\_complete: function(handler) {

this.get\_events().removeHandler("complete", handler);

},

// Create a function to raise the complete event.

raiseComplete: function() {

var h = this.get\_events().getHandler('complete');

if (h) h(this);

},

// Release resources before control is disposed.

dispose: function() {

var element = this.get\_element();

$clearHandlers(element);

Demo.Section.callBaseMethod(this, 'dispose');

}

}

Demo.Section.registerClass('Demo.Section', Sys.UI.Control);

// Since this script is not loaded by System.Web.Handlers.ScriptResourceHandler

// invoke Sys.Application.notifyScriptLoaded to notify ScriptManager

// that this is the end of the script.

if (typeof(Sys) !== 'undefined') Sys.Application.notifyScriptLoaded();