1

13

JavaScript: Events

2

The wisest prophets make sure of the event first.

— **Horace Walpole**

Do you think I can listen all day to such stuff?

— **Lewis Carroll**

3

The user should feel in control of the computer; not the other way around. This is achieved in applications that embody three qualities: responsiveness, permissiveness, and consistency.

— *Inside Macintosh, Volume 1*
Apple Computer, Inc., 1985

We are responsible for actions performed in response to circumstances for which we are not responsible.

— **Allan Massie**

4

OBJECTIVES

In this chapter you will learn:

- The concepts of events, event handlers and event bubbling.
- To create and register event handlers that respond to mouse and keyboard events.
- To use the event object to get information about an event.
- To recognize and respond to many common events.

5

Outline

6

Outline

registering.html

(1 of 3)

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 13.1: registering.html -->
6  <!-- Event registration models. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Event Registration Models</title>
10       <style type = "text/css">
11          div { padding: 5px;
12                margin: 10px;
13                border: 3px solid #0000BB;
14                width: 12em }
15       </style>
16       <script type = "text/javascript">
17          <!--
18          // handle the onclick event regardless of how it was registered
19          function handleEvent()
20          {
21             alert( "The event was successfully handled." );
22          } // end function handleEvent
23
24          // register the handler using the traditional model
25          function registerHandler()
26          {
27             var traditional = document.getElementById( "traditional" );
28             traditional.onclick = handleEvent;
29          } // end function registerHandler
```

```
30          // -->
31       </script>
32    </head>
33    <body onload = "registerHandler()">
34       <!-- The event handler is registered inline -->
35       <div id = "inline" onclick = "handleEvent()">
36          Inline registration model</div>
37
38       <!-- The event handler is registered by function registerHandler -->
39       <div id = "traditional">Traditional registration model</div>
40    </body>
41 </html>
```

Outline

registering.html

(2 of 3)

Outline

registering.html

(3 of 3)

Common Programming Error 13.1

Putting quotes around the function name when registering it using the inline model would assign a string to the `onclick` property of the node—a string cannot be called.

10

Common Programming Error 13.2

Putting parentheses after the function name when registering it using the inline model would call the function immediately and assign its return value to the `onclick` property.

11

Outline

onload.html

(1 of 2)

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 13.2: onload.html -->
6  <!-- Demonstrating the onload event. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>onload Event</title>
10       <script type = "text/javascript">
11          <!--
12          var seconds = 0;
13
14          // called when the page loads to begin the timer
15          function startTimer()
16          {
17             // 1000 milliseconds = 1 second
18             window.setInterval( "updateTime()", 1000 );
19          } // end function startTimer
20
21          // called every 1000 ms to update the timer
22          function updateTime()
23          {
24             ++seconds;
25             document.getElementById( "soFar" ).innerHTML = seconds;
26          } // end function updateTime
27          // -->
28       </script>
29    </head>
```

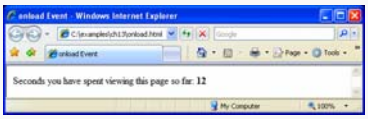

12

Outline

onload.html

(2 of 2)

```
30    <body onload = "startTimer()">
31       <p>Seconds you have spent viewing this page so far:
32       <strong id = "soFar">0</strong></p>
33    </body>
34 </html>
```



13

Common Programming Error 13.3

Trying to get an element in a page before the page has loaded is a common error. Avoid this by putting your script in a function using the `onload` event to call the function.

14

Outline

draw.html

(1 of 5)

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 13.3: draw.html -->
6  <!-- A simple drawing program. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Simple Drawing Program</title>
10       <style type = "text/css">
11          #canvas { width: 400px;
12                    border: 1px solid #999999;
13                    border-collapse: collapse }
14          td      { width: 4px;
15                    height: 4px }
16          th.key  { font-family: arial, helvetica, sans-serif;
17                    font-size: 12px;
18                    border-bottom: 1px solid #999999 }
19       </style>
20       <script type = "text/javascript">
21          <!--
22          //initialization function to insert cells into the table
23          function createCanvas ()
24          {
25             var side = 100;
26             var tbody = document.getElementById( "tablebody" );
27
```

15

Outline

draw.html

(2 of 5)

```
28             for ( var i = 0; i < side; i++ )
29             {
30                var row = document.createElement( "tr" );
31
32                for ( var j = 0; j < side; j++ )
33                {
34                   var cell = document.createElement( "td" );
35                   cell.onmousemove = processMouseMove;
36                   row.appendChild( cell );
37                } // end for
38
39                tbody.appendChild( row );
40             } // end for
41          } // end function createCanvas
42
43          // processes the onmousemove event
44          function processMouseMove( e )
45          {
46             // get the event object from IE
47             if ( !e )
48                var e = window.event;
49
50             // turn the cell blue if the Ctrl key is pressed
51             if ( e.ctrlKey )
52                this.style.backgroundColor = "blue";
53
54             // turn the cell red if the Shift key is pressed
55             if ( e.shiftKey )
56                this.style.backgroundColor = "red";
57          } // end function processMouseMove
```

```
58          // -->
59       </script>
60    </head>
61    <body onload = "createCanvas()">
62       <table id = "canvas" class = "canvas"><tbody id = "tablebody">
63       <tr><th class = "key" colspan = "100">Hold <tt>ctrl</tt>
64          to draw blue. Hold <tt>shift</tt> to draw red.</th></tr>
65       </tbody></table>
66    </body>
67 </html>
```

16

Outline

draw.html

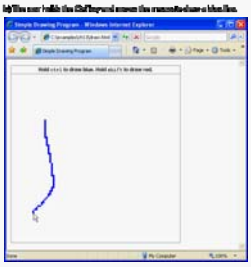
(3 of 5)

17

Outline

draw.html

(4 of 5)

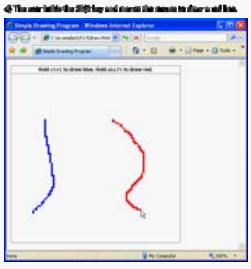


18

Outline

draw.html

(5 of 5)



19

Common Programming Error 13.4

Although you can omit the `tbody` element in an XHTML table, without it you cannot append `tr` elements as children of a `table` using JavaScript. While Firefox treats appended rows as members of the table body, Internet Explorer will not render any table cells that are dynamically added to a table outside a `thead`, `tbody` or `tfoot` element.

20

Property	Description
`altKey`	This value is `true` if the *Alt* key was pressed when the event fired.
`cancelBubble`	Set to `true` to prevent the event from bubbling. Defaults to `false`. (See Section 14.9, Event Bubbling.)
`clientX` and `clientY`	The coordinates of the mouse cursor inside the client area (i.e., the active area where the web page is displayed, excluding scrollbars, navigation buttons, etc.).
`ctrlKey`	This value is `true` if the *Ctrl* key was pressed when the event fired.
keyCode	The ASCII code of the key pressed in a keyboard event. See Appendix D for more information on the ASCII character set.
`screenX` and `screenY`	The coordinates of the mouse cursor on the screen coordinate system.
`shiftKey`	This value is `true` if the *Shift* key was pressed when the event fired.
`type`	The name of the event that fired, without the prefix `"on"`.

Fig. 13.4 | Some **`event`** object properties.

21

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 13.5: onmouseoverout.html -->
6  <!-- Events onmouseover and onmouseout. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Events onmouseover and onmouseout</title>
10       <style type = "text/css">
11          body  { background-color: wheat }
12          table { border-style: groove;
13                  text-align: center;
14                  font-family: monospace;
15                  font-weight: bold }
16          td    { width: 6em }
17       </style>
18       <script type = "text/javascript">
19          <!--
20          image1 = new Image();
21          image1.src = "heading1.gif";
22          image2 = new Image();
23          image2.src = "heading2.gif";
24
```

Outline

Onmouseoverout .html

(1 of 8)

```
25         function mouseOver( e )
26         {
27            if ( !e )
28               var e = window.event;
29
30            var target = getTarget( e );
31
32            // swap the image when the mouse moves over it
33            if ( target.id == "heading" )
34            {
35               target.src = image2.src;
36               return;
37            } // end if
38
39            // if an element's id is defined, assign the id to its color
40            // to turn hex code's text the corresponding color
41            if ( target.id )
42               target.style.color = target.id;
43         } // end function mouseOver
44
45         function mouseOut( e )
46         {
47            if ( !e )
48               var e = window.event;
49
50            var target = getTarget( e );
51
```


Outline

Onmouseoverout.html

(2 of 8)


```
52            // put the original image back when the mouse moves away
53            if ( target.id == "heading" )
54            {
55               target.src = image1.src;
56               return;
57            } // end if
58
59            // if an element's id is defined, assign id to innerHTML
60            // to display the color name
61            if ( target.id )
62               target.innerHTML = target.id;
63         } // end function mouseOut
64
65         // return either e.srcElement or e.target, whichever exists
66         function getTarget( e )
67         {
68            if ( e.srcElement )
69               return e.srcElement;
70            else
71               return e.target;
72         } // end function getTarget
73
74         document.onmouseover = mouseOver;
75         document.onmouseout = mouseOut;
76         // -->
77      </script>
78   </head>
```


Outline

Onmouseoverout.html

(3 of 8)


```
79   <body>
80      <img src = "heading1.gif" id = "heading" alt = "Heading Image" />
81      <p>Can you tell a color from its hexadecimal RGB code
82      value? Look at the hex code, guess its color. To see
83      what color it corresponds to, move the mouse over the
84      hex code. Moving the mouse out of the hex code's table
85      cell will display the color name.</p>
86      <table>
87         <tr>
88            <td id = "Black">#000000</td>
89            <td id = "Blue">#0000FF</td>
90            <td id = "Magenta">#FF00FF</td>
91            <td id = "Gray">#808080</td>
92         </tr>
93         <tr>
94            <td id = "Green">#008000</td>
95            <td id = "Lime">#00FF00</td>
96            <td id = "Maroon">#800000</td>
97            <td id = "Navy">#000080</td>
98         </tr>
99         <tr>
100           <td id = "Olive">#808000</td>
101           <td id = "Purple">#800080</td>
102           <td id = "Red">#FF0000</td>
103           <td id = "Silver">#C0C0C0</td>
104        </tr>
```


Outline

Onmouseoverout.html

(4 of 8)

```
105         <tr>
106            <td id = "Cyan">#00FFFF</td>
107            <td id = "Teal">#008080</td>
108            <td id = "Yellow">#FFFF00</td>
109            <td id = "White">#FFFFFF</td>
110         </tr>
111      </table>
112   </body>
113</html>
```

25

Outline

Onmouseoverout .html

(5 of 8)

26

Outline

Onmouseoverout .html

(6 of 8)

27

Outline

Onmouseoverout .html

(7 of 8)

28

Outline

Onmouseoverout.html

(8 of 8)

29

Performance Tip 13.1

Preloading images used in rollover effects prevents a delay the first time an image is displayed.

30

Outline

onfocusblur.html

(1 of 4)

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 13.6: onfocusblur.html -->
6  <!-- Demonstrating the onfocus and onblur events. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>A Form Using onfocus and onblur</title>
10       <style type = "text/css">
11          .tip { font-family: sans-serif;
12                 color: blue;
13                 font-size: 12px }
14       </style>
15       <script type = "text/javascript">
16          <!--
17          var helpArray =
18             [ "Enter your name in this input box.", // element 0
19               "Enter your e-mail address in this input box, " +
20               "in the format user@domain.", // element 1
21               "Check this box if you liked our site.", // element 2
22               "In this box, enter any comments you would " +
23               "like us to read.", // element 3
24               "This button submits the form to the " +
25               "server-side script.", // element 4
26               "This button clears the form.", // element 5
27               "" ]; // element 6
28
```

```
29         function helpText( messageNum )
30         {
31            document.getElementById( "tip" ).innerHTML =
32               helpArray[ messageNum ];
33         } // end function helpText
34         // -->
35      </script>
36   </head>
37   <body>
38      <form id = "myForm" action = "">
39         <div>
40         Name: <input type = "text" name = "name"
41            onfocus = "helpText(0)" onblur = "helpText(0)" /><br />
42         E-mail: <input type = "text" name = "e-mail"
43            onfocus = "helpText(1)" onblur = "helpText(0)" /><br />
44         Click here if you like this site
45         <input type = "checkbox" name = "like" onfocus =
46            "helpText(2)" onblur = "helpText(0)" /><br /><hr />
47
48         Any comments?<br />
49         <textarea name = "comments" rows = "5" cols = "45"
50            onfocus = "helpText(3)" onblur = "helpText(0)"></textarea>
51         <br />
52         <input type = "submit" value = "Submit" onfocus =
53            "helpText(4)" onblur = "helpText(0)" />
54         <input type = "reset" value = "Reset" onfocus =
55            "helpText(5)" onblur = "helpText(0)" />
56         </div>
57      </form>
```

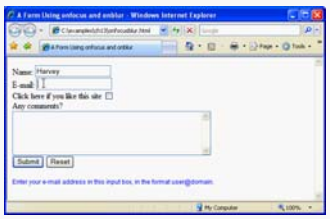

Outline

onfocusblur.html

(2 of 4)


```
58         <div id = "tip" class = "tip"></div>
59      </body>
60   </html>
```

b) The message changes depending on which field has focus. Now it gives instructions for the comments box.



Outline

onfocusblur.html

(3 of 4)

b) The message changes depending on which field has focus. Now it gives instructions for the comments box.

Outline

onfocusblur.html

(4 of 4)

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 13.7: onsubmitreset.html -->
<!-- Demonstrating the onsubmit and onreset events. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>A Form Using onsubmit and onreset</title>
      <style type = "text/css">
         .tip { font-family: sans-serif;
                color: blue;
                font-size: 12px }
      </style>
      <script type = "text/javascript">
         <!--
         var helpArray =
            [ "Enter your name in this input box.",
              "Enter your e-mail address in this input box, " +
              "in the format user@domain.",
              "Check this box if you liked our site.",
              "In this box, enter any comments you would " +
              "like us to read.",
              "This button submits the form to the " +
              "server-side script.",
              "This button clears the form.",
              "" ];

```

34

Outline

Onsubmitreset .html

(1 of 3)


```
         function helpText( messageNum )
         {
            document.getElementById( "tip" ).innerHTML =
               helpArray[ messageNum ];
         } // end function helpText

         function registerEvents()
         {
            document.getElementById( "myForm" ).onsubmit = function()
            {
               return confirm( "Are you sure you want to submit?" );
            } // end anonymous function

            document.getElementById( "myForm" ).onreset = function()
            {
               return confirm( "Are you sure you want to reset?" );
            } // end anonymous function
         } // end function registerEvents
         // -->
      </script>
   </head>
   <body onload = "registerEvents()">
      <form id = "myForm" action = "">
         <div>
         Name: <input type = "text" name = "name"
            onfocus = "helpText(0)" onblur = "helpText(0)" /><br />
         E-mail: <input type = "text" name = "e-mail"
            onfocus = "helpText(1)" onblur = "helpText(0)" /><br />
         Click here if you like this site
```

35

Outline

Onsubmitreset .html

(2 of 3)


```
         <input type = "checkbox" name = "like" onfocus =
            "helpText(2)" onblur = "helpText(0)" /><br /><hr />

         Any comments?<br />
         <textarea name = "comments" rows = "5" cols = "45"
            onfocus = "helpText(3)" onblur = "helpText(0)"></textarea>
         <br />
         <input type = "submit" value = "Submit" onfocus =
            "helpText(4)" onblur = "helpText(0)" />
         <input type = "reset" value = "Reset" onfocus =
            "helpText(5)" onblur = "helpText(0)" />
         </div>
      </form>
      <div id = "tip" class = "tip"></div>
   </body>
</html>
```

36

Outline

Onsubmitreset .html

(3 of 3)

Outline

bubbling.html

(1 of 3)

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 13.8: bubbling.html -->
6  <!-- Canceling event bubbling. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Event Bubbling</title>
10       <script type = "text/javascript">
11          <!--
12          function documentClick()
13          {
14             alert( "You clicked in the document." );
15          } // end function documentClick
16
17          function bubble( e )
18          {
19             if ( !e )
20                var e = window.event;
21
22             alert( "This will bubble." );
23             e.cancelBubble = false;
24          } // end function bubble
25
26          function noBubble( e )
27          {
28             if ( !e )
29                var e = window.event;
30
```


Outline

bubbling.html

(2 of 3)

```
31             alert( "This will not bubble." );
32             e.cancelBubble = true;
33          } // end function noBubble
34
35          function registerEvents()
36          {
37             document.onclick = documentClick;
38             document.getElementById( "bubble" ).onclick = bubble;
39             document.getElementById( "noBubble" ).onclick = noBubble;
40          } // end function registerEvents
41          // -->
42       </script>
43    </head>
44    <body onload = "registerEvents()">
45       <p id = "bubble">Bubbling enabled.</p>
46       <p id = "noBubble">Bubbling disabled.</p>
47    </body>
48 </html>
```

c) The user clicks the first paragraph, for which bubbling is enabled.

Outline

bubbling.html

(3 of 3)

b) The paragraph's event handler cancels the event.

c) The document's event handler also receives the event, because the event bubbles up to the document.

d) The user clicks the second paragraph, for which bubbling is disabled.

e) The paragraph's event handler cancels the event. The document's event handler is not called.

40

Common Programming Error 13.5

Forgetting to cancel event bubbling when necessary may cause unexpected results in your scripts.

41

Event	Description
`onabort`	Fires when image transfer has been interrupted by user.
`onchange`	Fires when a new choice is made in a `select` element, or when a text input is changed and the element loses focus.
`onclick`	Fires when the user clicks using the mouse.
`ondblclick`	Fires when the mouse is double clicked.
`onfocus`	Fires when a form element gains focus.
`onkeydown`	Fires when the user pushes down a key.
`onkeypress`	Fires when the user presses then releases a key.
`onkeyup`	Fires when the user releases a key.
`onload`	Fires when an element and all its children have loaded.
`onsubmit`	Fires when a form is submitted.
`onunload`	Fires when a page is about to unload.

Fig. 13.9 | Cross-browser events. (Part 1 of 2.)

42

Event	Description
`onmousedown`	Fires when a mouse button is pressed down.
`onmousemove`	Fires when the mouse moves.
`onmouseout`	Fires when the mouse leaves an element.
`onmouseover`	Fires when the mouse enters an element.
`onmouseup`	Fires when a mouse button is released.
`onreset`	Fires when a form resets (i.e., the user clicks a reset button).
`onresize`	Fires when the size of an object changes (i.e., the user resizes a window or frame).
`onselect`	Fires when a text selection begins (applies to `input` or `textarea`).
`onsubmit`	Fires when a form is submitted.
`onunload`	Fires when a page is about to unload.

Fig. 13.9 | Cross-browser events. (Part 2 of 2.)