JavaScript Events

hen you use an operating system such as Windows or Macintosh, you are using a Graphical User Interface (GUI). This interface reacts to mouse movement, clicking on icons, clicking of buttons, and much more. You can create the same effects with JavaScript by using event handlers. JavaScript event handlers are used to create menus, rollover buttons, dynamic content, and much more.

Event handlers are built-in methods that are attached to Web page elements that react to the user's actions. For example, you can use an event handler to detect if a user clicked on a button. You can use an event handler to change an image source to display a new image. Using JavaScript event handlers can turn a static Web page into a dynamic one.

JavaScript event handlers are bits of code that link a user's actions with the JavaScript code. The user actions can execute single or multiple lines of JavaScript statements including calling functions, changing the elements' properties, and changing the page's location. The actions that can be detected are clicking the mouse, pressing a keyboard key, selecting or changing form elements, or even loading and unloading a Web page. These are just a few examples. You can position JavaScript event handler statements within HTML tags just like the other attributes. You can set them equal to a small line of JavaScript code or to the name of a JavaScript function declared elsewhere in the Web page. When the element detects the specified event, the event handler executes the code.

For example, you can add the onclick event as an attribute to the
button> tag and set it equal to the function named blink, like this:

<button onclick="blink();">The Button</button>

This tag not only displays a button on the Web page, but after you click this button, the blink() JavaScript function is also executed.

There are a number of different events that you can use within a Web page. However, the places where you can use them are different for each event. Some of the more common JavaScript events are covered in this chapter, including mouse events, keyboard events, and selection events. These events create dynamic forms, menus, and effects that add to the user's experience at your Web site.

Mouse Events

Mouse events fire when the user clicks or moves the mouse. If the user clicks the mouse button once, the onclick event fires. If the user clicks the mouse button twice, the ondblclick event is fired.

For more precise details, you can use the onmousedown event to signal when the mouse button is pressed down and the onmouseup event when a mouse button is released. The onmouseover event detects when the mouse cursor moves over the top of an element. The onmouseout event detects when the mouse cursor moves off an element.

Mouse events are commonly used on images, divs, spans, and buttons to change the element's appearance and call JavaScript events. By adding an onclick event handler to a button, you can turn it into a link.

The onkeypress event is used to detect when a key on the keyboard is pressed. The specific key that was pressed is found in the window.event.keycode object.

Similar to the mouse button clicks, the onkeydown event detects after the key is first pressed and held down and the onkeyup event fires after the pressed key is released.

Keyboard events are commonly used in form fields to limit the keystrokes that can be entered. The keyboard events are also used to create shortcuts on navigation windows and hotkeys to perform specified JavaScript functions.

Selection Events

When an element is highlighted in the browser, it is said to have focus. The onfocus event is used to signal when an element has the focus. Pressing the Tab key or clicking on another element can change the element's focus. When an element loses the focus, the onblur event executes.

Another common event used with form elements is the onchange event. This event fires whenever the data of the form element is changed. This is commonly used on text boxes and selection lists.

Selection events are commonly used to detect changes on form elements to perform validation to verify that the information that is displayed is correct. Performing validation on the client side helps to eliminate bandwidth that a Web page can use and is another added layer in security.

Page Events

The onload event is used to detect when a Web page has completely finished loading. Similarly, the onunload event is fired when a Web page is unloaded. This happens when you leave the current page or when you click the browser's Refresh button.

When you cancel a Web page that is loading into your browser, then the onabort event is executed. When the user changes the width and height of the browser, the onchange event is fired.

Page events are used to initialize functions when something has happened to the browser whether it was opened, closed, stopped, or resized. The onload event is very common in most JavaScript applications to initialize a function when the page has been fully loaded. The onload event helps to eliminate errors by ensuring that all the necessary information is available to the JavaScript function.

Detect a Mouse Click

ou can interact with the user by detecting the mouse click on Web page elements. When the user clicks elements on the Web page, he or she usually expects an operation to happen. Buttons and links are the prime target for clicking, but other Web page elements may be clicked as well. You can use the onclick event handler to initialize a function, open a pop-up window, validate a form, show hidden content, change the page location, and much more.

You can detect a single mouse click by using the onclick event. The onclick event then triggers a JavaScript operation.

The onclick event is added to Web page elements by including it as an attribute to the element. A few examples of attributes are the src attribute in an image and the target attribute of a link. For example, when the user clicks on this image <img src="apple.jpg"

Detect a Mouse Click

onclick="alert('apple')">, an alert message appears
with the word "apple" in it.

If you want to detect double clicks of the mouse button, you can use the ondblclick event. This event fires when the mouse button is clicked twice in rapid succession.

The ondblclick event is added to the HTML element just like the onclick event. For example, you can add the ondblclick event to a button like this <button name="bl" value="Two" ondblclick="ClickMe()"> statement. When the button is clicked twice, the ClickMe() function is called.

You can add both onclick and ondblclick events in the same tag, but the onclick event fires every time the mouse button is clicked. Therefore, when the mouse is clicked twice, the onclick event fires twice, and the ondblclick event fires once.



	 The text displayed on the button has changed 	C\VisualBlueprint\EHAPTER_4\MouseElick.html - Microsoft Internet Explorer File Edit View Favorites Tools Help	_ & ×
	Dauble elisistic encoded butter	🕞 Back + 🕥 + 📓 🕼 🔎 Search 👷 Favorites 🔤 📻 + 📃	
8	Double-click the second button.	Address @ C:\VisuaBlueprint\CHAPTER_4\MouseClick.html	→ Go
		Clicked	-
	 An alert message appears. 	C:\VisualBlueprint\CHAPTER_4\MouseClick.html - Microsoft Internet Explorer	- 8 ×
9	Click the OK button to close the alert		140
	dialog box.	Address	-> Go
	The alert hox dialog hox closes		
		Clicked	
		Double Click	
		Microsoft Internet Evoluter	
		L Double Clicked	
		ок 🔫 9	
			-
		Done	
	Apply It		
	You can use the onclick event hand	dler to detect when the mouse button is clicked and	
	released, but you may want to detec	t each operation separately to perform separate	
	+ operations. You can detect the clicking	ng and releasing actions of the mouse button by using the	
	and holds it the anglish such that	ent nanciers. If a user clicks a mouse button on a page	

and holds it, the onclick event does not fire because it requires the mouse button to be released. The onmousedown function fires instead. You can use the following example to show the order in which the elements are fired when the HTML button is clicked.

Detect a Mouse Click

```
<form name="test">

<button name="B1" onclick="document.test.t1.value='Fired'"

onmousedown="document.test.t2.value='Fired'"

onmouseup="document.test.t3.value='Fired'">TEST</button><br>

<input type="text" name="t1">onclick<br>

<input type="text" name="t2">onmousedown<br>

<input type="text" name="t3">onmouseup

</form>
```

Create Rollover Buttons

o make your navigation more user friendly, you can use rollover buttons. When the mouse cursor moves over the button, the button changes its appearance. You can accomplish this by changing the source of an image or changing the Cascading Style Sheets' properties of the element. Cascading Style Sheets is referred to as CSS. You can learn more about CSS in Chapter 14.

You can create rollover buttons by using the onmouseover and onmouseout events. The events are added to the <input> tag of the element just like any attribute. For example, the onmouseover event in this statement displays A Dog in the status bar when the cursor is placed on top of the image.

The onmouseout function fires when the mouse leaves the image. A common mistake for programmers is to use the

term onmouseoff instead of onmouseout event. There is no event called onmouseoff so the code does not execute.

The onmouseover and onmouseout events are commonly used with links, buttons, text inputs, divs, and spans. The onmouseover and onmouseout events play a large role in many DHTML scripts as seen in Chapter 15.

You can change the source of an image when the mouse is placed on top of the image. To do this you need to change the image's source. For example the statement changes the image source to a new image when the cursor is over the image. If you want the image to change back to the original when the cursor leaves the image, then you need to add an onmouseout referencing the image's source.

A rollover button is turned into a link by using an onclick event handler to change the page's location.

Create Rollover Buttons



6	Open the HTML in your Web	
	browser.	

 The image displays the default source file set in the src attribute.

🚰 C:\VisualBlueprint\CHAPTER_4\rollover.html - Microsoft Internet Explorer	_ 8 ×
File Edit View Favorites Tools Help	
🕓 Back 🔻 🕥 👻 📓 🏠 🔎 Search 👷 Favorites 🛛 چ 📄 👻 🛄	
Address C:\VisualBlueprint\CHAPTER_4\rollover.html] → 60
HOME	Ă



8

Position the mouse cursor over the top of the image.

The image changes to the image specified in the onmouseover event.

</script>



Detect a Key Press

ou can detect when a user pushes a key on its keyboard while your Web site is in focus. You can use the onkeypress method to determine which key was pressed. By detecting the key press, you can develop hot keys, limit text entered into a text box, block key functions, and much more.

The actual key that was pressed is identified using the window.event.keycode object. The keycode object returns a number that corresponds to that key. For example, the keycode with the number 13 corresponds to the Enter key. In Appendix A, there is a table of the keycodes mapped to the corresponding keys.

You can also use two other key press events, onkeydown and onkeyup, which detect when the user holds a key down and releases it. The onkeypress event fires only when it detects a complete stroke of a key. The onkeydown event fires while the key is being pressed and the onkeyup event fires when the key that is being held down is released.

The key press events are added to multiple form elements on the Web page by setting the key press event as an attribute in the element's tag. The event is detected only when the element has focus. For example, the statement <input type="text" name="T1"

onkeypress="TypeFun()"> calls the function TypeFun() whenever a key is pressed while the text box T1 is in focus.

With the onkeypress event, you can assign all your interface elements a quick selection key to easily navigate through menus and forms. This enables users to navigate your interface using the keyboard instead of the mouse. It also makes your site accessible for people with disabilities.

Detect a Key Press



8 Open the HTML file in your Web browser.

Olick in the text field to give it focus.



- Enter a keyboard character.
 - An alert message appears giving the key code value of the button along with what key was pressed.
- Click the OK button to close the alert box.
 - You can continue to type characters into the text box to determine the key code.

C:\VisualBlueprint\CHAPTER_4\keyPress.htm - Microsoft Internet Explorer	_ <u> ×</u>
File Edit View Favorites Tools Help	RU.
🕝 Beck 🔻 🕥 👻 🛃 🏠 🔎 Search 🔮 Favorites 🛛 چ 🚍 🔹 🛄	
Address 🖉 C:\VisualBlueprint\CHAPTER_4\keyPress.htm	▼ 🗦 Go
A Microsoft Internet Explorer X 65 A OK	

Apply It

You can limit character input in text boxes and text areas by using the onkeypress event. Not all keys on the user's keyboard can be detected or cancelled out by using JavaScript. A common key that cannot be detected is the Print Screen key. Other keys like the function keys can be detected, but you may not be able to cancel their default action. For example, the F11 key causes a Web page to become a full screen. You can detect when the key is pressed, but you cannot cancel the event.

Limit Character Input

```
<script>
function handleKeyPress(evt) {
  var nbr, chr;
  if (window.Event) nbr = evt.which;
  else nbr = event.keyCode;
  if(nbr==13||nbr==12||nbr||11){//place key codes here
  return false;}
  }
  document.onkeydown= handleKeyPress
</script>
```

Detect a Modifier Key Combination

ou can detect when a user is holding down a combination key press with a modifier key. The modifier keys are the Alt, Ctrl, and Shift keys. These keys perform special browser functions. For example, the key combination of Ctrl+N opens up a new browser instance. The property names for the keys are altkey, ctrlkey, and shiftkey.

You can use the modifier key combinations to perform JavaScript tasks such as opening new browser windows, opening navigation menus, filling out forms, performing special tasks, changing the page's location and so on. You have to make sure that the key combination you pick does not already have a built-in function. For example, you cannot use the Ctrl+H combination because the predefined function opens the history menu or the Ctrl+B that is used to organize your Favorites. The default function fires when the pre-defined combination is selected. If you assigned the same combination in your code, the modifier combination may or may not be recognized depending on the browser.

Macintosh browsers have the same type of accelerator keyboard combinations as Windows. Instead of using the Ctrl key, the users use the meta key to perform their tasks. The property name of the meta key is metakey.

You can add a function for every key combination on the keyboard when designing your Web site; however, you should limit your choices so you do not confuse or overwhelm your users. Well-placed keyboard commands make your site's interface easier to use. You should use this as an alternative for navigating your Web site and not for the sole method of navigation.

Detect a Modifier Key Combination



Open the file in a Web	C:\VisualBlueprint\CHAPTER_4\modifierPress.htm - Microsoft Internet Explorer	_ 8 ×
browser.	File Edit View Favorites Tools Help	145
Press Shift+A.	Back + 🕑 - 💌 🖉 🎧 🔎 Search 🏠 Favorites 🔤 📼 -	
The taxt in the taxt have	Address 2:\VisualBlueprint\CHAPTER_4\modifierPress.htm	→ Go
reveals what keys were pressed.	shift 65	2
Press Ctrl+Shift+V	C:\VisualBlueprint\CHAPTER_4\modifierPress.htm - Microsoft Internet Explorer	_ 8 ×
	File Edit View Favorites Tools Help	<u>A</u> Y
 The text box reveals the new key combination 	🕞 Back + 🕥 - 💌 😰 🏠 🔎 Search 📌 Favorites 😓 🥽 +	
new key combination.	Address C:\VisualBlueprint\CHAPTER_4\modifierPress.htm	▼ → Go
		4
	cntrl shift 22	
Apply It		
+ + + + + + + + + + + + + + + + + + +	keys perform special JavaScript tasks. You can change the document location, dow, open a pop-up window, and more. Instead of adding the event handler to can assign the event to the entire document. For example, the following statemer = LoadValues executes the function LoadValues when the document loads.	nt
Assign an Event		
function handle	KevPress(evt) {	
var nbr;		
if (window	.Event) nbr = evt.which;	
else nbr =	e vent.keyCode;	
if(nbr==72){document.location.href="http://www.wiley.com";}	
else if(nk	<pre>sr==80) {window.open("http://www.JavaRanch.com");}</pre>	
else{retur	n true;}	
} document.onkevdowr	n= handleKeyPress	
You can alter this scri	pt to add more else-if statements to perform more tasks. You can also add	

multiple JavaScript statements in each if statement to perform multiple tasks per key press. By setting the JavaScript statement to return false;, you can keep the key press from happening.

Set and Remove Focus

ou can place the cursor in any text field on the page by using the focus() method. For example, the statement <body onload="document.test.text1. focus()"> places the cursor in text box named text1 when the page loads. This method is great for pages that have a login box, because it eliminates the need for the user to click the box, thus saving the user time.

You can detect when a Web page element receives focus by using the onfocus event. The onfocus event fires when the user starts to interact with the element. By using the onfocus() event, you can change the background color of a form element. For example, the statement <input type="text" name="T1" onfocus="this.style. background='yellow'"> changes the background color of the textbox T1 when the element is focused. If you want to remove focus from a Web page element, you can use the blur() method. The blur() method removes the focus from the element. Therefore, if the user is typing in a text box, the cursor is removed and the user can no longer type in the box. You can change the background of the element to another color after the focus has been removed from the element. For example, the statement <input type="text" name="1" onblur="this.style. background='red'"> changes the background color of the textbox T1 when the element's focus is removed.

When the user removes focus of an element by clicking on another part of the page or using the tab key to advance to the next element, you can detect it by using the onblur event. The onblur event is often used to validate a form element to make sure that the validation parameters are met. You can use these methods and events on frames, text boxes, text areas, pop-up windows, and the parent page.





- The text updates according to the fields on focus event.
- Click another text field.
- 🚰 C:\¥isualBlueprint\CHAPTER_4\focusText.html Microsoft Internet Explorer _ 8 × R File Edit View Favorites Tools Help 2 Search 5 🔶 Back 👻 θ × Favorites - 1 🔹 🄁 Go Address @ C:\VisualBlueprint\CHAPTER_4\focusText.html Hello I have focus 🚰 C:\VisualBlueprint\CHAPTER_4\focusText.html - Microsoft Internet Explorer _ 8 × File Edit View Favorites Tools Help A. C) 🗲 Back 👻 × Search Favorites 200 -1 💌 🌛 Go Address C:\VisualBlueprint\CHAPTER_4\focusText.html l lost focus I have focus function MoveFocus(name) { if (document.form1.T1.value.length == 3 && name=='T1') {; document.form1.T2.focus();
 - The onblur event fired when the text box lost focus changing the text.
 - The onfocus fired in the other text field after it gained focus.
- Note: The onblur event will fire when it loses focus.

Apply It

You can move focus from element to element on a form to make it easier for a user to fill out the form. This eliminates the need for a user to have to move the mouse or use the Tab key. The following function MoveFocus () is called by the text box. You can add an onkeydown () to watch what the user types into the form fields you want. You can monitor more text boxes by adding more if statements into the function.

```
Move Focus
<script>
```

```
</script>
<form name="form1">
    <input type="text" name="T1" onkeydown="MoveFocus('T1')">
    <input type="text" name="T2" >
</form>
```

Onchange Event

ou can detect when the user has changed its selection in a drop-down list or even the text in a textbox with the onchange event. The onchange event is triggered when the user removes the focus from the form element. If the value of the form element is different than when the user first interacted with the element, the onchange event executes. If the value is the same as the initial value, then the onchange event handler is ignored.

The onchange event is added to the form element just like the attributes. For example, the statement <input type="text" name="T1" onchange="Validate()"> calls the JavaScript function Validate() when the text is changed.

The onchange event handler helps to eliminate any unnecessary steps that the user would have to take when visiting your Web site. Common uses for using the onchange event are validating forms, submitting forms, changing the page location, and calling functions without any additional user input.

A common place for the onchange event handler is with a select drop-down list. When the onchange event handler is added to the element, it is executed when a different option is selected from the list. A common script that uses the selection list with the onchange event is a navigation menu. When the option is selected in the menu, the page's location is changed. The onchange event handler eliminates the need for the user to click a button to change the page's location.

When using the onchange event with a drop-down list, it is common for the user to reselect the original value. The onchange event does not execute because the value has not changed. If you still need to detect that the user removed the focus from the element, you should use the onblur event.

Onchange Event





Click	the first option in the	C:\VisualBlueprint\CHAPTER_4\onchange.html - Microsoft Internet Explorer	_8×
selec	tion list.	File Edit View Favorites Tools Help	<u>R</u>
•	The text field updates	🕞 Back 👻 🛞 🖌 😰 🏠 🔎 Search 👷 Favorites 🛛 🌺 🚍 🕶 🛄	
á	according to the	Address 🖉 C:\VisualBlueprint\CHAPTER_4\onchange.html	▼ → Go
	onchange event .	Pencil Paper Pen Ruler Cost \$0.99	~
Click	the second option in the	C:\VisualBlueprint\CHAPTER_4\onchange.html - Microsoft Internet Explorer File Edit View Eavorites Tools Help	_ B ×
Click selec	the second option in the tion list.	C:\VisualBlueprint\CHAPTER_4\onchange.html - Microsoft Internet Explorer File Edit View Favorites Tools Help Back	_ & ×
Click selec	the second option in the stion list.		
Click selec	the second option in the tion list. The text field updates again.	Image: Stress	
Click selec	t the second option in the ction list. The text field updates again.	C:\VisualBlueprint\CHAPTER_4\onchange.html - Microsoft Internet Explorer File Edit View Favorites Tools Help Back Image: Search Ima	
Click selec	t the second option in the option list. The text field updates again.	Image: Section of the section of th	
Click selec	the second option in the ction list. The text field updates again.	C:\VisualBlueprint\CHAPTER_4\onchange.html - Microsoft Internet Explorer File Edit View Favorites Tools Help Back Image: Search Image: Search Image: Favorites Image: Search	
Click selec	the second option in the option list. The text field updates again.	Image: Section of the section of th	

You can use the onchange () event to validate form field elements. You can detect when the user has changed the information and call a function to verify the input. The example code checks to see if all the parameters are met. If the fields are not met, then the form value is reset and the field is refocused after an alert dialog box message is displayed. If the user does not change the value, the function does not fire again. Therefore, you may want to use the onblur () event. The onblur () event forces the user to have the correct information entered.

```
Validate Form Fields
<script>
    function Validate() {
         if(document.form1.T1.value.length < 5){
             alert('5 or more characters required');
              document.form1.T1.value = "";
             document.form1.T1.focus();
    }
</script>
<form name="form1">
    <input type="text" name="T1" onchange="Validate()">
</form>
```

Handle the Page Load and Onunload Operations

ou can detect when a Web page has fully loaded by using the onload event handler. The onload event is normally placed as an attribute inside the <body> tag.

When the browser has loaded the last element on the Web page, the onload event is triggered. The onload event handler is typically used to initialize events after the page loads. This eliminates the chance of an error caused by the JavaScript function trying to use an object on the Web page that has not loaded. It is common to see DHTML scripts use the onload event to generate dynamic HTML elements on the page.

A similar event to the onload event is the onunload event. This event executes when the current Web page is exited. For example, the onunload event will fire for browser closings, back button, refresh button, and using links. The onload and onunload are well known because many Web sites abuse them by opening pop-up advertisements on their Web site when you enter or leave a site.

There is no cross browser method to detect when a browser is closed. However, an Internet Explorer only event can capture the closing of the browser. The onbeforeunload event executes when exiting a page, prompting the user whether to continue ones action. There are some elaborate scripts available online that try to detect the browser's closing. Either these scripts do not work with every browser's or they increase the page loading time significantly.

You can only have one onload and onunload event handler on the page. Chapter 17 discusses ways to eliminate the problems with multiple onload and onunload event handlers.

📕 loadMessages.html - Notepad Add the onload event to the File Edit Format View Help <body> tag. <html> <head> Set the onload event to display </head> welcome message. 2 <body ="document.form1.T1.value<u>=</u>'Page Fully Loaded"-3 Add the onunload event to the onunload <body> tag. <form name="form1"> <input type="text" name="T1" value="Page Loading"> Set the onunload event to </form> display goodbye message. </body> Save the file. /html>

Open the file in a Web browser.

 The text box displays the loading message. Wait until Page Fully Loaded appears in the text box.

C:\VisualBlueprint\CHAPTER_4\loadMessages.html - Microsoft Internet Explorer	_ 8 ×
File Edit View Favorites Tools Help	AU.
🕞 Back + 🕑 + 😰 🕼 🔎 Search 👷 Favorites 🔤 🖃 + 🛄	
Address 🖉 C:\VisualBlueprint\CHAPTER_4\JoadMessages.html	▼ → Go
Page Loading	×

Handle the Page Load and Onunload Operations



Extra

You can detect a few more browser events including when the stop button is clicked and when the page is resized. You can use the onabort event to detect when the stop button is pressed on the browser. The onabort event is used to notify the user that the page needs to be fully loaded in order for your code to work. You add the onabort event handler as an attribute within the <body> tag. You can use the onresize event to detect when the user has changed the dimensions of its browser screen. Just like the onabort event, you add this code as an attribute in the <body> tag.

Detect Browser Events

```
<html>
<head>
<title> JavaScript </title>
<script>
function ChangeSize(){
window.resizeTo(300,300);
}
</script>
</head>
<body onResize="ChangeSize()">
Make Page Remain 300px X 300px.
</body>
</html>
```



ou can use an HTML link to execute a JavaScript statement or to call a function. Using links is a good way to call a JavaScript function since a user is used to clicking on them. You are also able to use CSS properties from Chapter 14 to format the links so they change properties when the mouse is over them.

There are two major ways to use a link. The first method is to add the JavaScript statement inside the href attribute of the tag. First, you need to add the javascript: keyword inside the tags, followed by colons, and then the JavaScript statement. For example, the statement Test is a link that calls the function named Test when the link is clicked. You can add multiple JavaScript statements by placing them after the semicolon of each statement. The second way that you can execute JavaScript statements from an HTML link is by attaching the onclick event handler to the object. You need to add a return false; statement to the end of the parameters that the onclick handler executes. This stops the link from performing its operation. You also need to include a value in the href attribute. If you do not include an href attribute, then the link does not appear as a link in the browser. For example, the statement Test executes the function Test when the link is clicked.

Instead of including a hash sign (#) in the href attribute of the link, you can add a page that notifies the user that the Web page requires JavaScript to be enabled. The user is redirected to that Web page since the browser ignores the onclick handler when JavaScript is disabled.

Execute a JavaScript Statement from an HTML Link

Add the javascript: keyword to the href attribute in the link.
 Insert the JavaScript statement after the javascript: keyword.
 Save the file.
 Open the file in a Web browser.
 Click the link to execute the JavaScript statement.
 The link executes the JavaScript statement displaying an alert dialog box.



Attach an Event to an Object

ou can eliminate the need of adding the same exact event handler to every element on the page by attaching the event handler to an object. This eliminates the need to type hundreds of statements when building functions to perform validation checks or mouseover effects.

Event handlers are attached to the document, window, or any specific objects. Specific objects can include the <body> tag, text boxes, select boxes, radio buttons, and so on. A common event handler that is used with the window object is the onload event handler. The statement window.onload = Start; attaches the onload event handler to the window object and calls the start function.

You can bind the event handler to the object by first referencing the object. Then, add the event handler after the reference and set the statement equal to the function that should be called. For example, the statement document.form1.T1.onblur = Validate; binds the onblur event handler to the form element name T1.

By using a DOM method explained in Chapter 15, you can simplify the process by looping through all the elements and adding the event handler. Therefore, you do not have to type out hundreds of statements adding the event handler. The JavaScript function can do it for you. You can find all the elements on the page by using the getElementsByTagName() and setting the parameter inside the parentheses to the tag name. For example, the statement var theTags = document. getElementsByTagName("input"); places all the input tags into an array which you can loop through and attach the event handler. When looping through the function, you can attach the event handler with the statement theTags[i].onchange= setCount;. The event handler is added to all the tags when the array is iterated through.

Attach an Event to an Object



Determine the Element That Received the Event



ou can detect what element received the event handler and apply this to a validation function or any other function. This method eliminates the need to have to add the element reference to each tag. Instead, you can attach the events through a JavaScript statement. This eliminates the need to type a lot of information that may be added dynamically to the page.

Microsoft Internet Explorer differs from the DOM event detection, which means you need to do some detection. Internet Explorer uses the srcElement event while DOM uses the target event to determine where the event came from. The statement elem = evt.target ? evt.target : ((evt.srcElement) ? evt.srcElement : null); determines what event the browser needs to use to detect the element. Another problem that you face is that some other browsers such as Netscape 6 use another method to determine the element. These browsers require that you look at the parent node to find the correct element. If the statement if (elem.nodeType == 3) is true, then you know you have one of these special cases. You then need to use the parentNode property to find the element.

After you determine that the browser needs to use srcElement or target events, you can detect all the properties of the element. For example, the statement var theValue = elem.value; retrieves the value of a form element that was triggered by the event handler.

A popular reason for using this script is for form validation. You can add an onblur event handler to your elements and have it checked to make sure it is correct. If not, you can have the element stay focused by using the statement elem.focus();.

Determine the Element That Received the Event

- Add a statement that handles browser compatibility for event handling.
- 2 Add code to determine the source element.
- 3 Display the element information on the page.

- 4 Add an onfocus event handler to a form element.
- 5 Repeat step **4** for each element of the form.

Save the file.



Open the file in a Web browser.
 Click a form element to set the object's focus.

• The properties of the element that is in focus display on the screen.



Extra

You can detect where the mouse came from on a page by assigning onmouseover and onmouseout objects to the page. This allows you to determine what area of the page the mouse has come from, thereby allowing you to call functions accordingly.

Detect Mouse

The code shows where the cursor was moved from on the page. It uses the div ids to find the necessary information. You can use any other attribute of that form to perform any action you want.

Determine Which Mouse Button Was Pressed

ou can differentiate between what mouse button was clicked by monitoring the onmousedown event handler. The onclick and onmouseup event handlers do not always catch the event. Detecting the mouse button allows you to perform different actions.

The button event is an integer number that corresponds to what button was pressed on the mouse. For Internet Explorer and the DOM, the numbers of the events differ. Internet Explorer uses the number one for the left button, which is also considered the primary button. The DOM uses the number zero to correspond to the left button. The middle button on the mouse is number four for Internet Explorer and number one for DOM. The right button is number two for both Internet Explorer and DOM. When no buttons are pressed, Internet Explorer returns the number zero while DOM returns null. Internet Explorer also detects other combination of button presses that DOM does not. A left and right button click corresponds to number three. The left and middle combination corresponds to number five, while a right and middle click correspond to number six. The last combination is a left, middle, and right button click that corresponds to the integer number seven.

0

When you are programming, you need to recognize that not everyone has three buttons on its mouse. Most people have two buttons. It is also common for Macs to have only one button on their mouse. Mac users have the ability to right-click objects by holding down a modifier key on the keyboard, but not everyone knows how to do this. Limiting dependability on button detection makes your script more operating system friendly.

Determine Which Mouse Button Was Pressed



Cancel Browser Events

ou may want to block a user from triggering an event on the page. For example, one thing that you can block is the contextual menu, which is better known as the right-click menu. Before you block an event, you need to consider the benefit that you are giving the user by blocking the event. If you are not giving them a benefit, then you may be wasting your time and irritating the user that is visiting your site. Some people use the right mouse button as a way to navigate through a page, so by blocking this you may be alienating visitors of your site.

You can cancel an event by using the cancelBubble event. When the cancelBubble event is set to true, the event does not fire. You also need to return false at the end of the function to make sure the event is fully blocked. Not returning false allows the action to still perform. By assigning an event at the document level, you can block most events. The event to block the right click button is to use the oncontextmenu event. By setting this to the document object and calling a function, you can disable the context menu.

If you are using this to block users from saving images, you need to remember that images are being downloaded to the user's hard drive. The images are automatically saved in the Temporary Internet Files folder. The person may be about to hot link to an image to download it. The user can disable JavaScript to download the image, use print screen, or save the document with the Save As function or a site ripper program. A determined user can steal your images or view your source code.

Cancel Browser Events

- Determine the source of the event accessing the function.
- Verify that the event was executed while accessing an image.
- 3 Set the cancelBubble method to true and add a return false statement to block the action fully.
- 4 Add a return true statement for all other events.
- Attach the oncontextmenu to the document object assigning a function name.
- 6 Save the file.
 - Open the file in a Web browser.
- 8 Right-click the mouse on top of the image.
 - The oncontextmenu browser event was blocked.



Wy Comput