One additional jQuery function that can come in handy when queuing up effects on an element is the `delay()`. This function simply waits the specified number of milliseconds before beginning the next effect in the queue. For example, say you want to fade an image into view, wait 10 seconds, and then fade it out of view. You can use the `delay()` function like this:

```
$('#photo').fadeIn(1000).delay(10000).fadeOut(250);
```

Tutorial: Animated Dashboard

In this tutorial, you'll use the `animate()` function to move a `<div>` tag right from off the left edge of the page into view. The div is absolutely positioned (see the box on page 187 for more on absolute positioning) so that most of the box hangs off the left edge of the page outside the boundaries of the browser window (Figure 6-3, left). When a visitor mouses over the visible edge of the div, that div moves completely into view (Figure 6-3, right). To make this effect more fun, use the jQueryUI library to animate the background color of the div and to use a couple of different easing methods.

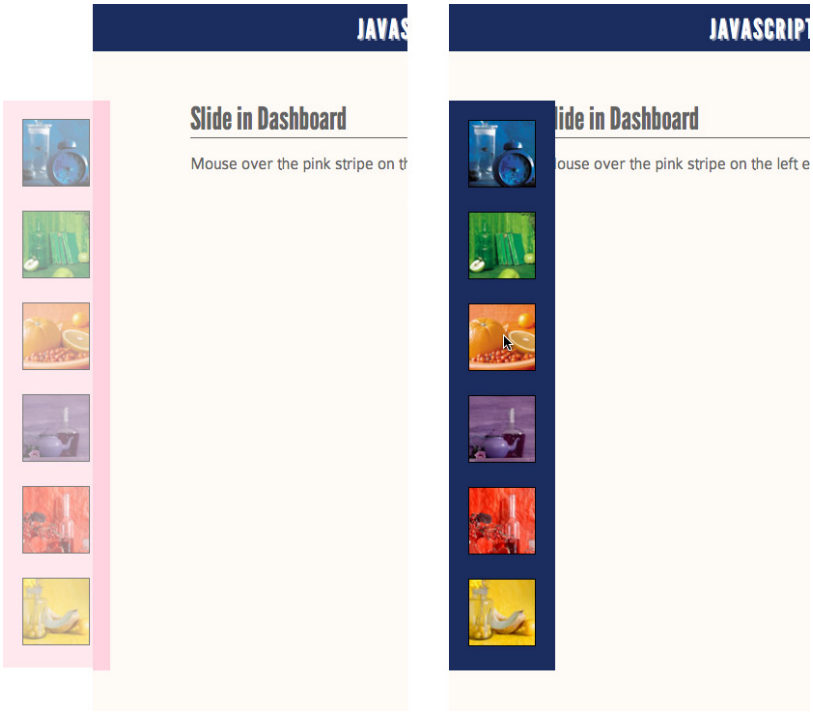

FIGURE 6-3
You can have a lot of fun by hiding page elements off one of the edges of the browser window (like the div in the left image, which is mostly out of view). Using the `animate()` function, you can then bring that page element fully into view (right).

NOTE See the note on page 12 for information on how to download the tutorial files.

The basic task is rather simple:

1. **Select the `<div>` tag.**

   Remember that a lot of jQuery programming begins with selecting an element on the page—in this case, the `<div>` tag that a visitor mouses over.

2. **Attach a `hover` event.**

   The `hover` event (described on page 162) is a special jQuery function, not a real JavaScript event, that lets you perform one set of actions when a visitor mouses over an element, then a second set of actions when the visitor mouses off the element (the `hover` event is really just a combination of the `mouseEnter` and `mouseLeave` events).

3. **Add the animate function for the `mouseEnter` event.**

   When a visitor mouses over the div, you'll animate the left position of the div, moving it from out of view on the left edge of the browser window. In addition, you'll animate the background color of the div.

4. **Add another animate function for the `mouseLeave` event.**

   When a visitor moves the mouse off the div, you'll animate the div back to its original position and with its original background color.

The Programming

1. **In a text editor, open the file *animate.html* in the *chapter06* folder.**

   This file already contains a link to the jQuery file, and the `$(document).ready()` function (page 160) is in place. However, because you'll be animating the background color of the div and using a couple of interesting easing methods, you need to attach the jQueryUI file.

2. **Click in the empty line after the first `<script>` tag and add the code in bold below:**

```
<script src="../_js/jquery-1.7.2.min.js"></script>
<script src="../_js/jquery-ui.min.js"></script>
```

   jQueryUI is a jQuery *plug-in*. In the jQuery world, a plug-in is simply an external JavaScript file that lets you add complex effects and features to your site without a lot of programming on your part. Next you'll select the div and add the `hover()` function to it.

3. **Click in the empty line inside the `$(document).ready()` function and type `$('#dashboard').hover(); // end hover` so your code looks like this:**

```
$(document).ready(function() {
  $('#dashboard').hover(); // end hover
}); // end ready
```

`$('#dashboard')` selects the `<div>` tag (which has the ID dashboard applied to it). The `hover()` function takes two arguments—two functions (page 162)—that describe what to do when a visitor moves his mouse over the div, and then moves his mouse off the div. Instead of typing all of the code at once, you'll build it up piece by piece, first adding the `hover()` function, then adding the "shell" for two anonymous functions. This approach is helpful because the nest of parentheses, braces, commas, and semicolons can overwhelm you if you're not careful.

4. **Click between the parentheses of the `hover()` function and add two empty, anonymous functions:**

```
$(document).ready(function() {
  $('#dashboard').hover(
   function() {

   },
   function() {

   }
  ); // end hover
}); // end ready
```

jQuery's `.hover()` function is a bit wild looking when you've finished adding all of its programming (see step 12). However, it's really just a function that accepts two functions as arguments. It's a good idea to add the "shells" of the anonymous functions first—so you can make sure you've set it up correctly—before adding all of the programming inside the functions.

TIP Test your code frequently to make sure you haven't made any typos. In step 4, you can type `console.log('mouseEnter')` inside the first anonymous function and `console.log('mouseLeave')` inside the second anonymous function, and then preview the page in a web browser. You'll need to open the browser's console to see the results: F12 for Internet Explorer; Ctrl+Shift+J (Win) or ⌘-Option-J (Mac) for Chrome; Ctrl+Shift+K (Win) or ⌘-Option-K (Mac) for Firefox; and, ⌘-Option-C for Safari. The console should spit out the message "mouseEnter" when you mouse over the div, and "mouseLeave" when you mouse out. If no message appears in the console, you've made a typo. Double-check your code against these steps, or use the steps on page 18 to find errors using the browser's error console.

5. **Inside the first anonymous function, type `$(this).animate(); // end animate`.**

As discussed on page 159, inside an event, `$(this)` refers to the page element to which you've attached the event. In this case, `$(this)` refers to the `<div>` tag with the ID dashboard. In other words, mousing over this div will also animate this div.

6. **Add an object literal with the CSS properties you wish to animate:**

```
$(document).ready(function() {
  $('#dashboard').hover(
   function() {
```

```
    $(this).animate(
  {
    left: '0',
    backgroundColor: 'rgb(27,45,94)'
  }
    ); // end animate
  },
  function() {

  }
 ); // end hover
}); // end ready
```

The first argument to the `animate()` function is an object (page 136) containing CSS properties. In this case, the div currently has a left value of -92px, so that most of the div is hidden, hanging off the left edge of the browser window. By animating its left value to 0, you're essentially moving it completely into view on the left edge. Likewise, thanks to jQueryUI, you're changing its background color from pink to dark blue. Next, you'll set a duration for the animation.

7. **Type a comma after the closing } of the object literal, press Return and then type 500.**

   The comma marks the end of the first argument passed to the `animate()` function, while the 500 sets the length of the animation to half a second or 500 milliseconds. Lastly, you'll set an easing method.

8. **Type a comma after the 500, hit Return, and type `'easeInSine'` so your code looks like this:**

```
$(document).ready(function() {
  $('#dashboard').hover(
   function() {
     $(this).animate(
       {
         left: '0',
         backgroundColor: 'rgb(27,45,94)'
       },
       500,
    'easeInSine'
     ); // end animate
   },
   function() {

   }
  ); // end hover
}); // end ready
```

The last argument to the `animate()` function here—`'easeInSine'`—tells the function to use an easing method that starts off somewhat slowly and then speeds up.

9. **Save the file. Preview it in a browser and mouse over the div.**

   The div should scoot into view. If it doesn't, troubleshoot using the techniques described on page 18. Of course, when you mouse off the div, nothing happens. You have to add the animate function to the second anonymous function.

10. **Add the code below to the second anonymous function:**

```
$(this).animate(
    {
       left: '-92px',
       backgroundColor: 'rgb(255,211,224)'
    },
    1500,
    'easeOutBounce'
); // end animate
```

   This code reverses the process of the first animation, moving the div back off the left edge of the window and reverting the background color to pink. The timing is a bit different—1.5 seconds instead of half a second—and the easing method is different.

11. **Save the file. Preview it in a browser and move your mouse over and off of the div.**

   You'll see the div move into view and then out of view. However, if you move the mouse over and off the div repeatedly and quickly, you'll notice some strange behavior: The div will keep sliding in and out of view long after you've finished moving the mouse. This problem is caused by how jQuery queues up animations on an element. As described on page 194, any animation you add to an element gets put into a sort of queue for that element. For example, if you have an element fade into view, fade out of view, and then fade back into view, jQuery performs each effect in order, one after the other.

   What's happening in the code for this tutorial is that each time you mouse onto and off of the div, an animation is added to the queue; so, rapidly mousing over the div creates a long list of effects for jQuery to perform: Animate the div into view, animate the div out of view, animate the div into view, animate the div out of view, and so on. The solution to this problem is to stop all animations on the div before performing a new animation. In other words, when you mouse over the div, and that div is in the process of being animated, then jQuery should stop the current animation, and proceed with the animation required by the `mouseEnter` event. Fortunately, jQuery supplies a function—the `stop()` function—for just such a problem.

12. **Add `.stop()` between `$(this)` and `.animate` in the two anonymous functions. The finished code should look like this (additions are in bold):**

```
$(document).ready(function() {
  $('#dashboard').hover(
   function() {
     $(this).stop().animate(
       {
         left: '0',
         backgroundColor: 'rgb(27,45,94)'
       },
       500,
       'easeInSine'
     ); // end animate
   },
   function() {
     $(this).stop().animate(
       {
         left: '-92px',
         backgroundColor: 'rgb(255,211,224)'
       },
       1500,
       'easeOutBounce'
     ); // end animate
   }
  ); // end hover
}); // end ready
```

The `.stop()` function here simply ends any animations on the div before starting a new one, and prevents multiple animations from building up in the queue.

Save the page and try it out in a web browser. You can find a finished version of this tutorial—*complete_animate.html*—in the *chapter06* folder.

jQuery and CSS3 Transitions and Animations

If you're keeping up with the latest and greatest in Cascading Style Sheets techniques, you may be wondering why even bother with jQuery? After all, CSS transitions let you use CSS alone to produce animations between two different CSS styles, and CSS animations let you produce complex animated effects (see Figure 6-4).

The new CSS animation effects are awesome, but not all browsers can enjoy them. Two still widely used versions of Internet Explorer—8 and 9—don't support either CSS transitions or animations. So, if you need to include those browsers in your website building plans, you'll need another way to add animation. In this instance,