5. Set up event listeners for the `input` event of each of the range sliders. The code is going to set the `filter` and the `-webkit-filter` properties on the image in the `<figure>`. Recall from Chapter 7 that if you are setting multiple filters, they have to be included together separated by spaces.
6. Add a listener for the click event of the reset button. This will simply remove the filters from the image.

Testing

1. To test, click on the thumbnails and verify the correct caption is displayed. Ensure the filters work as expected.

PROJECT 2: Travel

DIFFICULTY LEVEL: Intermediate

HANDS-ON EXERCISES

Project 10.2

Overview

This project will build a photo gallery using jQuery for our travel photo sharing site as shown in Figure 10.22.

Instructions

1. Examine **lab10-project2.html** in the browser and then editor. You have been supplied with the appropriate CSS (the relevant classes are in `gallery.css`), html, and JavaScript data files (an array of image objects are in `images.js` file). The data is minimized in that file so there is an additional file called `data.json` which contains the data in an easy-to-read format. The images are supplied in two folders: `images/square` (for the gallery) and `images/medium` (for the popup).
2. Loop through the images array and using the appropriate jQuery DOM methods, add the appropriate `<img>` tags to the supplied `<ul class="gallery">` element. The image filenames are contained in the `path` property of each image object. Set the `alt` attribute of each `<img>` to the `title` property of the image object.
3. Use jQuery to attach handlers for the `mouseenter`, `mouseleave`, and `mousemove` events of the square images in the gallery.
4. For the `mouseenter` event, use jQuery to add the `"gray"` class to the square `<img>` under the mouse. If you examine that class, you will see it sets the `filter` property to `grayscale()`. Hint: remember that `$(this)` within an event handler references the DOM object that generated the event.
5. Also for the `mouseenter` event, use jQuery to generate a `<div>` with an `id="preview"` (the styling for `#preview` is already defined in `gallery.css`). Within that `<div>` add an `<img>` element that displays the larger version of the image. Underneath that `<img>` add a `<p>` element for the caption. The information for the caption and image are contained within the `images` array. The `alt` attribute of the square image under the mouse contains the image

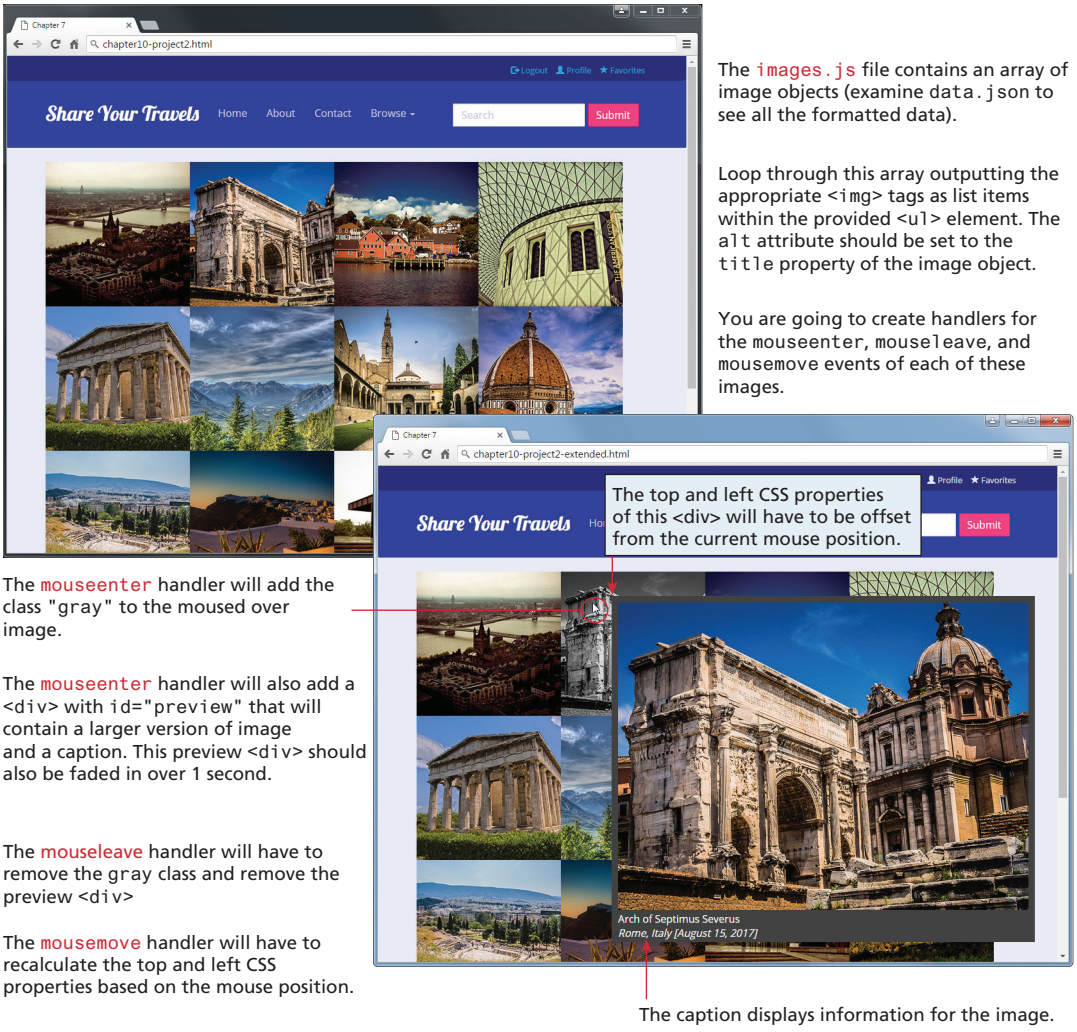

FIGURE 10.22 Project 2

title. You can search through the `images` array looking for a match on the title; once a match is found, you have the file path, city, country, and date information.

6. You will need to use jQuery to set the `left` and `top` CSS properties for the `#preview<div>`. You can retrieve the x, y coordinates (via the `pageX` and `pageY` properties) of the current mouse position from the event object that is passed to your event handler. You can calculate the new position by offsetting by some amount from the mouse x, y position.

7. Finally, once the `#preview <div>` is constructed, simply append it to the `<body>`.
8. For the `mouseleave` event, remove the `"gray"` class from the square image under the mouse. Also remove the `#preview<div>` from the body.
9. For the `mousemove` event, simply set the `left` and `top` CSS properties for the `#preview <div>` using the same approach as described in step 6.

Testing

1. Verify the code works when mousing over the images. Be sure that the caption is displaying the correct information.
2. Don't worry if the pop-up image is "off screen" when mousing over images on the edges of the browser.

PROJECT 3: CRM Admin

DIFFICULTY LEVEL: Advanced

HANDS-ON EXERCISES

Project 10.3

Overview

This project will use jQuery AJAX to consume and display JSON data. It will also make use of a third-party JavaScript library to display charts of that data (see Figure 10.23).

Instructions

1. Examine **lab10-project3.html** in the browser and then editor. You have been supplied with the appropriate CSS files as well. This project can be a bit overwhelming, so we advise breaking it down into smaller steps: at each step below, test to ensure it works.
2. First, you will populate the `#filterBrowser <select>` list with a list of browsers. The data for this list is going to be retrieved using the `$.get()` method. The URL for the web service is as follows:

   http://www.randyconnolly.com/funwebdev/services/visits/browsers.php

   You may want to first examine the JSON that is returned (simply by entering this URL into a browser window). You will notice it returns an array of objects: each object contains the browser `id` and `name`.

   Now you want to programmatically retrieve this information using the `$.get()` method. When the data is retrieved (i.e., within the `done()` handler) you will loop through the returned data and add an `<option>` element to the `#filterBrowser <select>` list for each browser in the returned data. Be sure to set the `value` attribute for each `<option>` to the `id` property (e.g., `<option value="2">Chrome</option>`).
3. Do the same for the countries and operating system <select> lists. The URL for the operating system list web service is as follows:

   http://www.randyconnolly.com/funwebdev/services/visits/os.php