The CSS is

```
p {  position: absolute;
     left: 200px;
     top: 100px;
     font-family: Arial, sans-serif;
     width: 300px; }
```

The HTML source code is

```
<h1>Absolute Positioning</h1>
   <p>This paragraph is 300 pixels wide and uses CSS absolute
   positioning to be placed 200 pixels in from the left and 100 pixels
   down from the top of the browser window.</p>
```

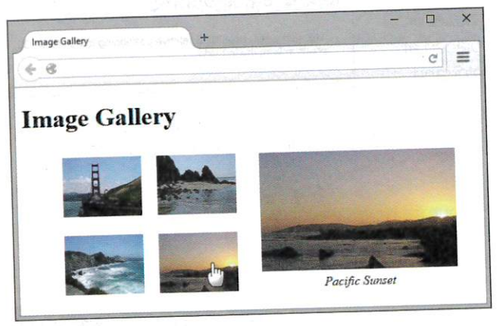


**Figure 6.45** An interactive image gallery with CSS

## Practice with Positioning

Recall that the CSS `:hover` pseudo-class provides a way to configure styles to display when the web page visitor moves the mouse over an element. You'll use this basic interactivity along with CSS positioning and display properties to configure an interactive image gallery with CSS and HTML. Figure 6.45 shows the interactive image gallery in action (see chapter6/6.9/gallery.html in the student files). When you place the mouse over a thumbnail image, the larger version of the image is displayed along with a caption. If you click on the thumbnail, the larger version of the image displays in its own browser window.

## Hands-On Practice 6.9

In this Hands-On Practice, you will create the interactive image gallery web page shown in Figure 6.45. Copy the following images located in the student files chapter6/starters folder into a folder named gallery2: photo1.jpg, photo2.jpg, photo3.jpg, photo4.jpg, photo1thumb.jpg, photo2thumb.jpg, photo3thumb.jpg, and photo4thumb.jpg.

Launch a text editor and modify the chapter2/template.html file to configure a web page as indicated:

1. Configure the text, Image Gallery, within an h1 element, and within the title element.
2. Code a div assigned to the id named `gallery`. This div will contain the thumbnail images, which will be configured within an unordered list.
3. Configure an unordered list within the div. Code four li elements, one for each thumbnail image. The thumbnail images will function as image links with a `:hover` pseudo-class that causes the larger image to display on the page. We'll make this all happen by configuring an anchor element containing both the thumbnail image and a span element that comprises the larger image along with descriptive text. An example of the first li element is



```
<li><a href="photo1.jpg"><img src="photo1thumb.jpg" width="100"
   height="75" alt="Golden Gate Bridge">
   <span><img src="photo1.jpg" width="250" height="150"
   alt="Golden Gate Bridge"><br>Golden Gate Bridge</span></a>
</li>
```

4. Configure all four li elements in a similar manner. Substitute the actual name of each image file for the href and src values in the code. Write your own descriptive text for each image. Use photo2.jpg and photo2thumb.jpg in the second li element. Use photo3.jpg and photo3thumb.jpg in the third li element. Use photo4.jpg and photo4thumb.jpg for the fourth li element. Save the file as index.html in the gallery2 folder. Display your page in a browser. You'll see an unordered list with the thumbnail images, the larger images, and the descriptive text. Figure 6.46 shows a partial screen capture.

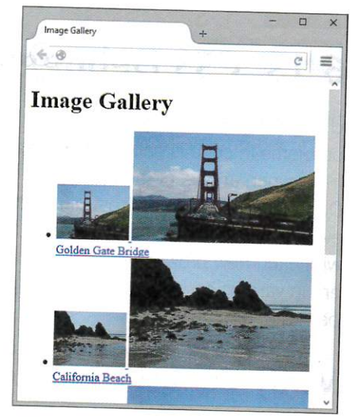


**Figure 6.46** The web page display before CSS

5. Now, let's add embedded CSS. Open your index.html file in a text editor and code a style element in the head section. The `gallery` id will use relative positioning instead of the default static positioning. This does not change the location of the gallery but sets the stage to use absolute positioning on the span element in relation to its container (`#gallery`) instead of in relation to the entire web page document. This won't matter too much for our very basic example, but it would be very helpful if the gallery were part of a more complex web page. Configure embedded CSS as follows:

   a. Set the `gallery` id to use relative positioning.
   ```
   #gallery { position: relative; }
   ```
   b. The unordered list in the gallery should have a width of 250 pixels and no list marker.
   ```
   #gallery ul { width: 250px; list-style-type: none; }
   ```
   c. Configure the list item elements in the gallery with inline display, left float, and 10 pixels of padding.
   ```
   #gallery li { display: inline; float: left; padding: 10px; }
   ```
   d. The images in the gallery should not display a border.
   ```
   #gallery img { border-style: none; }
   ```
   e. Configure anchor elements in the gallery to have no underline, #333 text color, and italic text.
   ```
   #gallery a { text-decoration: none; color: #333;
                font-style: italic; }
   ```
   f. Configure span elements in the gallery not to display initially.

```
#gallery span { display: none; }
```

g. Configure the span elements in the gallery to display *only* when the web visitor hovers the mouse over the thumbnail image link. Set the location of the span to use absolute positioning. Locate the span 10 pixels down from the top and 300 pixels in from the left. Center the text within the span:

```
#gallery a:hover span { display: block; position: absolute;
        top: 10px; left: 300px; text-align: center; }
```

Save your page and display it in a browser. Your interactive image gallery should work well in modern browsers. Compare your work to Figure 6.45 and the sample in the student files (chapter6/6.9/gallery.html).

# 6.13 CSS Debugging Techniques

Using CSS for page layout requires some patience. It takes a while to get used to it. Fixing problems in code is called **debugging**. This term dates back to the early days of programming when an insect (a bug) lodged inside the computer and caused a malfunction. Debugging CSS can be frustrating and requires patience. One of the biggest issues is that even modern browsers implement CSS in slightly different ways. Browser support changes with each new browser version. Testing is crucial. Expect your pages to look slightly different in various browsers. The following are helpful techniques to use when your CSS isn't behaving properly:

## Verify Correct HTML Syntax

Invalid HTML code can cause issues with CSS. Use the W3C Markup Validation Service at http://validator.w3.org to verify the correct HTML syntax.

## Verify Correct CSS Syntax

Sometimes a CSS style does not apply because of a syntax error. Use the W3C CSS Validation Service at http://jigsaw.w3.org/css-validator to verify your CSS syntax. Carefully check your code. Many times, the error is in the line *above* the style that is not correctly applied.

## Configure Temporary Background Colors

Sometimes your code is valid but the page is not rendered in the way that you would expect. If you temporarily assign distinctive background colors such as red or yellow and test again, it should be easier to see where the boxes are ending up.

## Configure Temporary Borders

Similar to the temporary background colors, you could temporarily configure an element with a 3 pixel red solid border. This will really jump out at you and help you recognize the issue quickly.