The `set_names` function is responsible for reading the XML returned by the server and building the results. The JavaScript in the `set_names` function isn't particularly hard; it's just tedious and an unnatural way to write HTML content. Remember, though, that building the `onmouseover`, `onmouseout`, and `onclick` event handlers required nonstandard code; otherwise, it wouldn't work in all major browsers. Had you not known the workaround, you may not have figured out why the example worked in some browsers but not others.

Now, compare the code in Listing 8-13 with the code in Listing 8-14, which shows `searchAutocomplete.jsp`. Listing 8-14 demonstrates how you build the content of the results drop-down list using the Taconite framework.

Listing 8-14. `searchAutocomplete.jsp`

```jsp
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://taconite.sf.net/tags" prefix="tac" %>

<tac:taconiteRoot>
  
  <tac:replaceChildren contextNodeID="popup" parseOnServer="true">
    
    <c:forEach var="result" items="${results}"
      
      <div onmouseover="hilite(this);" onmouseout="unhilite(this);">
        
        <a href="${result.url}" class="autocomplete" target="_blank">
          ${result.title}
        </a>
      
      </div>
    
    </c:forEach>
  
  </tac:replaceChildren>

</tac:taconiteRoot>
```

The code in Listings 8-13 and 8-14 performs the same task. Each variant builds the results that appear in the results drop-down list. (The JavaScript in Listing 8-13 places each result in a table row, and the JSP in Listing 8-14 places each result in a `div` element.) Not only is the JSP/Taconite method slightly shorter (18 lines versus 23 lines), but you'll surely agree that the code in Listing 8-14 looks much more natural and easier to write. You'll appreciate writing HTML markup using HTML markup rather than creating it programmatically through JavaScript.

Better yet, remember those pesky workarounds for the `onmouseover`, `onmouseout`, and `onclick` event handlers? In `searchAutocomplete.jsp`, the event handlers are written as simple attributes on the HTML elements. Taconite automatically generates cross-browser JavaScript to create these event handlers, freeing you from the responsibility of remembering what the various workarounds are.

The search component contains a number of enhancements over the autocomplete example from Chapter 4. The Chapter 4 autocomplete example performed an Ajax request every time the user typed a character into the text box. People who type fast create a higher frequency of Ajax requests, increasing the load on the server with little benefit for the user, since rapid typing leaves little time between keystrokes to display the updated results.