while (document.getElementById("weatherContent").childNodes.length > 0) {
    document.getElementById("weatherContent").removeChild(
        document.getElementById("weatherContent").childNodes[0]);
}
document.getElementById("weatherContent").appendChild(element0);
</taconite-replace-children>
</taconite-root>

Note how the XML begins with a taconite-root tag that has two taconite-replace-children tags, which is similar to the structure of the tags in Listing 8-7. Each taconite-replace-children tag has a CDATA section that contains the embedded JavaScript to appropriately update the page's content. If you carefully read the embedded JavaScript in Listing 8-8, you'll easily be able to see exactly what the JavaScript is doing. In fact, it's likely identical to the JavaScript you would write to update the page! Upon receiving this XML response from the server, the Taconite browser-side library automatically extracts and executes the embedded JavaScript using the eval function.

Analyzing the Headline News Component

The headline news component is the simplest component of the Ajax Dashboard. The component polls the server at set intervals and updates its contents with the current headline news stories. Unlike the other three components, the news component never sends any data to the server. Instead, it simply asks for the most current news items and displays them. Since no data is being sent to the server, the amount of code needed to actually send the request is ridiculously small, as shown here:

function updateNewsItems() {
    var ajaxRequest = new AjaxRequest("UpdateNewsItems");
    ajaxRequest.sendRequest();
}

This component is implemented slightly differently than the weather forecast component. Both rely on the Taconite framework to dynamically create content that is specified within the body of a JSP. The weather forecast component uses Taconite's JSP tags to build the response that is returned to the server. The news component, in contrast, eschews the use of Taconite's JSP tags and instead relies on the developer to build the XML response that is returned to the client. In this scenario, the parsing of the HTML to JavaScript will occur within the browser rather than on the server.

What are the benefits of avoiding the use of Taconite's JSP tags? The main benefit is that this approach demonstrates how the Taconite framework can be server-side technology agnostic. As long as the tools used on the server can generate an XML document, you can use this approach along with the Taconite framework.

A side benefit of using the browser-side parser instead of the server-side parser is lower bandwidth usage. When using the JavaScript parser, the desired HTML content is returned to the browser as just that—HTML. When using the server-side Java parser, the HTML content is translated into several JavaScript commands that are returned to the browser to be executed.