Articles and tutorials continue to pop up weekly if not daily. While following some of the sites already listed will tip you off to most if not all of these works, it’s worthwhile to peruse O’Reilly (www.oreilly.com) and the Apple Developer Connection (developer.apple.com). Both sites have been used as resources for this book and provide timely information on a range of Ajax topics.

**Using a Framework**

Throughout this book, we’ve given you a fair number of Ajax examples. And the astute observer may have noticed a fair amount of, well, duplicate code. For instance, how many times have you seen something like Listing 8-1?

**Listing 8-1. The Most Repeated Chunk of Code in This Book**

```javascript
var xmlHttp;

function createXMLHttpRequest() {
    if (window.ActiveXObject) {
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    } else if (window.XMLHttpRequest) {
        xmlHttp = new XMLHttpRequest();
    }
}
```

Of course, in a production application, we’d abstract this little routine. Actually, we’d probably go further and create a special library that encapsulated the messy, repetitive aspects of Ajax. Then again, we might do a quick Google search and discover quite a few hits for Ajax frameworks. For a snapshot of available frameworks, check out Appendix B.

**Introducing Taconite**

It’s shameless-plug time: the authors of this book are also the co-creators of the open-source Ajax framework Taconite. We admit this is a bit self-serving, but seriously, we think Taconite is pretty sweet. While originally built for Java Enterprise Edition–based applications, we have refactored the core of Taconite into a client-side library that can easily be used with any server-side technology. Beyond that, it wouldn’t be terribly hard to port the Taconite server-side components to other technologies such as .NET.

What makes Taconite so special? Ajax is a fantastic step forward in the evolution of the Web application. However, we’ve struggled over the years with the inconsistencies among browsers and the difficulty inherent in developing massive quantities of JavaScript. Since we’re basically lazy, we decided to just “build it once” so we could easily reuse our hard work. (Besides, we can tell our bosses it’ll take three weeks, get it done in one, and spend the rest of the time following the trials and tribulations of the Minnesota Vikings.)

---

2. Seriously, how many times? Well, we counted: including this one, 19.