At the Forge

Dojo

Reuven M. Lerner

Abstract

Become a black belt in JavaScript in your very own Dojo.

JavaScript has experienced a renaissance in the past year or two. Whereas many Web developers long saw JavaScript as a second-class programming language, useful for (at best) decorating Web pages, it is an increasingly central technology for Web developers. Whether you are adding Ajax (Asynchronous JavaScript and XML), dynamic HTML or new GUI widgets to your Web pages, you likely have begun to use JavaScript more in the past year or two than ever before.

Luckily for all Web developers, the rapid and widespread interest in JavaScript programming has resulted in the development of JavaScript libraries and toolkits, many released under an open-source license. In my last few columns, we looked at Prototype, which aims to make general JavaScript programming easier, and at Scriptaculous, which provides visual effects and interface widgets. Prototype has become quite popular among open-source programmers, in no small part because of its inclusion in the Ruby on Rails application framework.

But, Prototype and Scriptaculous are far from the only games in town. Another popular open-source JavaScript framework is Dojo. Dojo is based on a number of convenience classes and objects begun by Alex Russell of JotSpot, a startup purchased by Google in 2006. Russell continues to work on Dojo, but contributions of code and money now come from other sources as well, including companies such as Sun and IBM. Moreover, Dojo is now included by default in the popular Django Web application framework, giving it additional exposure.

This month, we take an initial look at Dojo, examining the way it divides code into packages, then at several of the convenience functions it provides for JavaScript programmers and, finally, at a very small sample of Dojo's large widget library. Even if you have no intention of using Dojo, I hope you find this article instructive. I almost always find it useful to see how other languages and toolkits do things, if only to get some better perspective on what I am doing.

Installing Dojo

The first thing to understand about Dojo is that it is large, at least by JavaScript standards. (Remember that all JavaScript code must be downloaded from the server, interpreted by the browser and then executed within memory, all as quickly as possible. A large JavaScript library might offer many features, but it will make performance unacceptably slow.) Thus, although we might consider Dojo to be a single, large library, it is actually a collection of many smaller parts.

This is a relevant point even before you download Dojo, because the download site requires that you choose which combination of features you prefer to use. Knowing that my server is on a relatively high-speed line, that my sites tend to be relatively lightweight and that I plan to explore Dojo as a developer, I installed the “everything” version, labeled as kitchen sink on the download site. But, if you are interested in Dojo solely for its rich-text editor, or for use in Ajax or charting, you might want to download one of the many smaller versions, each identified by the subset of Dojo's functionality it covers. For the purposes of this column, however, I assume you also have downloaded the kitchen sink version.

At the time of this writing, Dojo is at version 0.4.1, and the kitchen sink version is available from the URL http://download.dojotoolkit.org/release-0.4.1/dojo-0.4.1-kitchen_sink.tar.gz.

Once you have downloaded the file, unpack it:

```
tar -zxvf dojo-0.4.1-kitchen_sink.tar.gz
```

The directory that you open will contain a number of different items, including a README file, several Flash animation (*.swf) files used in Dojo's persistent storage engine, the main dojo.js JavaScript file and several subdirectories, including one containing demos (called demos), one containing the source code for much of Dojo's functionality, and release and tests directories for development of the toolkit itself.

To get Dojo up and running on your server, you must put dojo.js and the src subdirectory under your document root. I tend to put my Web sites under /var/www/SITENAME/www, and JavaScript files go in the javascript directory under that path. I created a further subdirectory named dojo and put both dojo.js and the src directory there as well. Thus, the full path to dojo.js on my filesystem is /var/www/SITENAME/www/javascript/dojo/dojo.js, but the URL that we will use to load it from a Web page will be /javascript/dojo/dojo.js.

And, indeed, we can load Dojo into our Web pages using the standard <script> tag:

```
<script type="text/javascript"
src="/javascript/dojo.js"></script>
```

Although the above loads dojo.js into the browser's memory, this does not mean all of Dojo's commands are now available. Rather, including dojo.js makes it possible for us to load one or more of Dojo's individual packages.   You can think of dojo.js as a bootloader, in that its only purpose is to make Dojo available to you later on, rather than to perform any tasks on its own.

Dojo Packages

As we saw during the past few months, Prototype and Scriptaculous have fairly well-defined roles, and thus, they remain separate products. Prototype provides a large number of convenience functions for JavaScript programmers, and Scriptaculous adds GUI-related functionality on top of it. Dojo is designed with a different organizational philosophy in mind, providing a wide array of different functions, many of which might seem unrelated to one another.

For example, Dojo provides GUI elements (for example, a rich-text editor, a date picker, interfaces to mapping sites and layout containers). But, it also provides an event system, making it possible to assign functionality to particular events, using a variety of different models. It provides a client-side storage system with more sophistication than HTTP cookies. It provides a number of utilities for JavaScript programmers, making it possible to create new classes, send notes to a debugger or otherwise work with the language.

Each of these pieces of functionality is available inside of a separate package, which is both loaded and identified with a hierarchical name structure. Thus, all Dojo functions begin with dojo (for example, dojo.declare and dojo.debug), and they are loaded as part of a similarly named hierarchy.

Loading a Dojo package is as simple as putting:

```
<script type="text/javascript">
dojo.require("dojo.PACKAGE.*");
</script>
```

inside your HTML. You can load more than one package, using multiple invocations of dojo.require. Dojo's package loader is smart enough to take care of any dependencies that might exist.

JavaScript Helpers

Once you have included Dojo, you can begin to use some of its improvements to the JavaScript language.   Dojo includes a number of convenience functions to make JavaScript programming easier, some of which are quite similar to what Prototype offers. For example, nearly every JavaScript program needs to retrieve nodes based on their id attributes. (An id attribute is supposed to be unique in a particular page of HTML, thus allowing us to identify a node uniquely.) To assign the variable myNode to the node with the ID of target, we normally would need to write:

```
var myNode = document.getElementById("target");
```

Dojo allows us to abbreviate this to:

```
var myNode = dojo.byId("target");
```

This is not quite as short as Prototype's $() operator, but it is still a significant improvement, making programs both shorter and more readable.

Dojo also provides some new mechanisms to work with arrays and other enumerated lists. For example, it provides a foreach loop:

```
dojo.lang.forEach(arrayName, iterationFunctionName);
```

The above code causes iterationFunctionName to be invoked once for each element of arrayName. Thus, we could say:

```
var names = ["Atara", "Shikma", "Amotz"];
dojo.lang.forEach(names, alert);
```

to print each of these names in an alert box. Dojo provides several other convenient functions for use with arrays, including dojo.map (which invokes an operation on each element of an array, producing a new array as a result) and dojo.filter (which returns an array of those items for which a function returns true). Stylistically, the documentation for Prototype seems to encourage users to write inline functions, whereas the Dojo documentation encourages users to write named functions and then refer to them. However, you can adopt whichever style is more appealing to you.

Rich Text

One of the easiest parts of Dojo to begin using is its collection of widgets. From the time that HTML forms were first standardized, Web developers have wanted a richer set of widgets from which to choose, in order to provide applications that resemble—in style, as well as power—parallel widgets available for desktop applications. Dojo provides a number of such widgets, making it possible to include rich-text editors, sliders and combo boxes in our programs.

For example, we might want to use the Dojo rich-text editor, allowing people to write using more than the plain text that a <textarea> tag provides. We can do that simply by creating a <div> and giving it a class of dojo-Editor:

```
<div class="dojo-Editor">
    Hello from the Dojo editor!
</div>
```

If you fire up the above, you'll get...nothing, other than a <div> with some text inside of it, as you might expect without installing Dojo. This is because of Dojo's modular loading scheme; loading dojo.js is only the first step in using any of Dojo's functionality, bootstrapping the loading system. Loading the actual editor code requires that we invoke the dojo.require JavaScript function:

```
<script type="text/javascript">
dojo.require("dojo.widget.Editor");
</script>
```

Once we have done this (producing the file shown in Listing 1), we suddenly have a rich-text editor at our disposal. This is wonderful, except for one thing—how do we submit the HTML-formatted file to an application on our server? One method would be to use Ajax to save the contents of our div on a regular basis, submitting its contents to the server without any need for explicit saving. And, indeed, this is what many Web-based applications, including word processors and spreadsheets, have done. No longer do you need to save documents; you simply work with them, and you can expect the computer to save what you've done reliably.

Listing 1. simple.html *9554l1.qrk*

```
<html>
    <head>

    <title>Dojo page title</title>
        <script type="text/javascript"
        src="/javascript/dojo.js"></script>

        <script type="text/javascript">
            dojo.require("dojo.widget.Editor");
        </script>
    </head>

    <body>
          <div class="dojo-Editor">
              editor goes here
          </div>
    </body>
</html>
```

This month, we take a simpler approach, including our rich-text editor in an HTML form submission. Unfortunately, it's still not obvious how we can pull this off, because HTML forms consist of <input> and <textarea> tags. Luckily, the Dojo team has taken this into consideration. Notice that we define a Dojo widget by its class, not by its tag type. This means we can attach the dojo-Editor class to anything, not only to an empty <div> tag. If we attach it to a <textarea>—which is a block element, just like <div>—our text editor will be attached to a textarea, which will be submitted to the server. In other words, we replace our <div> with:

```
<textarea name="text" class="dojo-Editor">
    Hello from the Dojo editor!
</textarea>
```

Listing 2 shows an example of how this might look when incorporated into a simple HTML form. When the contents of the form are sent to the server, all formatting is preserved using HTML tags. Your application will need to parse this HTML to understand any formatting that might appear in the text.

Listing 2. simple-form.html *9554l2.qrk*

```
<html>
    <head>

    <title>Dojo page title</title>
        <script type="text/javascript"
        src="/javascript/dojo.js"></script>

        <script type="text/javascript">
            dojo.require("dojo.widget.Editor");
        </script>
    </head>

    <body>
       <form method="POST" action="/parse.php">
           <textarea name="text" class="dojo-Editor">
               write something!
           </textarea>
         <input type="submit" />
       </form>
    </body>
</html>
```

Of course, if your plan is to take the input text and simply display it in a Web browser, not much (if any) work is needed on your part. You can stick the input into a database and then retrieve it whenever it is needed. (I haven't checked into the security of this widget to make sure it is immune to cross-site scripting attacks, so you might want to investigate it further before simply accepting, storing and displaying user data.)

Conclusion

As you can already see, Dojo offers a wide variety of functions and doesn't take much effort to start using. But using many of the other widgets Dojo includes, such as an attractive DatePicker, requires that we use Dojo's sophisticated event handler, which we did not examine here. Next month, we will look at events in Dojo and how that package lets us incorporate special effects, Ajax and many more widgets into our Web applications.

Resources *9554s1.qrk*

The main source for information about Dojo, as well as Dojo software releases, is at http://dojotoolkit.org. Documentation for the toolkit is still a bit sparse, but it has improved significantly in the last few months, and continued improvements seem likely, given Dojo's growing popularity. The main URL for Dojo documentation is at http://dojotoolkit.org/docs, with Dojo.book (the Wiki-based Dojo documentation) at http://manual.dojotoolkit.org/index.html.

Some good articles about JavaScript toolkits, including Dojo, are at http://www.sitepoint.com/article/javascript-library.

Finally, a good introduction to rich-text editing with Dojo is at http://dojotoolkit.org/docs/rich_text.html.