Dojo Events and Ajax

Reuven M. Lerner

Abstract

The quality of your Dojo depends upon your connections.

Last month, we began looking at Dojo, one of the most popular open-source JavaScript toolkits to emerge in the last year or two. Although using a toolkit is not required if you want to include Ajax or sophisticated client-side functionality in your Web application, it certainly makes things a great deal easier. In particular, such toolkits typically know how to handle the many subtle differences in JavaScript implementations on different browsers. JavaScript is far more standardized than used to be the case, but a number of traps still exist when trying to work with multiple platforms, and using a toolkit can relieve you of having to handle them yourself.

In last month's article, we looked at Dojo's packaging system, some of its enhancements to the JavaScript language and its rich-text editor. This month, we look at some of Dojo's other capabilities that might interest modern Web developers, including support for events and Ajax.

Event Handlers

One of the cornerstones of JavaScript programming is the use of event handlers—functions that are invoked when a particular event occurs. For example, we can define a function that opens an alert box:

```<script type="text/javascript"> function openAlert() { alert("Hello! This is an alert!"); } </script>
```

We can then tell the user's browser to invoke our openAlert function whenever someone clicks on a paragraph of text:

```<p onclick="openAlert();">This is a paragraph.</p>
```

There are several interesting things to notice in this short example. First, we have set the onclick event handler in this case. About a half-dozen other event handlers exist from which we could choose. In many cases, we might set more than one event handler. This was particularly prevalent in the pre-CSS days, when JavaScript event handlers would be used to change the look of an icon when the mouse was hovering over it.
Second, event handlers sometimes can be used in contexts you might not expect. For example, the above `<p>` tag has an onclick handler. You normally wouldn't think of clicking on a paragraph of text, but we can do that. This is the basis for some of the modern drag-and-drop events.

Third, although JavaScript does make it pretty easy to attach handlers to particular events, some messiness still is involved. We cannot define multiple event handlers easily or disconnect handlers that have been defined.

**Dojo Events**

*By this point, you might be wondering what JavaScript event handlers have to do with using Dojo for Ajax and modern Web applications. The answer is that much of Dojo's functionality, across all of its many packages, depends on the event system. If you want to use Dojo's Ajax package, for example, you need to connect it using Dojo events. This might seem restrictive at first glance; however, Dojo events are surprisingly easy to understand.*

As a simple example, let's see how we might implement our onclick handler from before using Dojo events. First, we need to modify our event-handling function so that it takes one argument, the event itself:

```javascript
function openAlert(evt) {
  alert("Hello! This is an alert from Dojo!");
}
</script>
```

Next, we must connect the paragraph to the event. Rather than doing this directly, by setting the onconnect handler, we give our paragraph an id tag:

```html
<p id="para">This is a paragraph.</p>
```

Now, we can use Dojo's dojo.byId function—similar in some ways to Prototype's $() function—to get the node itself:

```javascript
var para = dojo.byId("para");
```

Finally, we connect our paragraph to the handler function we created:

```javascript
dojoevent.connect(para, "onclick", openAlert);
```

If we put it all together, we get the program shown in Listing 1, which I have called test-dojo.html.

**Listing 1. test-dojo.html**

```html
<html>
<head>
  <title>Testing JavaScript with Dojo</title>
  <script type="text/javascript">
    function openAlert(evt) {
      alert("Hello! This is an alert from Dojo!");
    }
  </script>

  <p id="para">This is a paragraph.</p>

  var para = dojo.byId("para");

  dojoevent.connect(para, "onclick", openAlert);

</head>
</html>
```
One thing you might notice is the three <script> tags in the file. The first, in the head of the document, downloads dojo.js, the main Dojo source file, from the server. The second, also in the head of the document, imports the Dojo package for events and defines our event-handling function, openAlert. The third and final piece of JavaScript, which attaches the node to the event, is in the body of the document, right after our p tag is defined. This is because we can set an event handler only for an object that already exists, which means after the p tag itself.

If you load the page into a browser window, you will see that it works just like the previous version. Given that this version is more complex, it might not seem obvious how it is better.

Advanced Dojo Events

Here, then, is one example. Suppose you want to invoke one particular object method, rather than a simple function call, in an event handler. JavaScript makes it difficult to do this directly from an event handler. However, dojo.event.connect handles this quite simply, in its four-parameter version. As before, the first two parameters are the node and event that will trigger the handler. The third and fourth arguments are the object and function that will be invoked. For example:

    dojo.event.connect(eventObject, "onclick", handlerObject, "handlerMethod");

Dojo also makes it possible to connect more than one handler to an event. In non-Dojo JavaScript, you could accomplish this only by making your event handler a function that then invokes other functions. Using Dojo events, you can connect any number of methods:

    Garrick, small font below.

    dojo.event.connect(para, "onclick", testFormContents);
    dojo.event.connect(para, "onclick", submitFormContents);

Events are fired in the order that they are connected. So, in the above example, testFormContents would be invoked before submitFormContents.

Note that Dojo allows you to add the same event handler twice, if you want. So, be careful to invoke
dojo.event.connect only once for each event-handler combination to avoid potentially odd and hard-to-debug problems.

Let's say you want to provide an expert mode to your users, so they don't have to see all of the annoying alert boxes we're generating. We could create a button that, when pressed, removes the event handler from the object—ooh, but now that's getting kind of tricky, especially if we have multiple events to deal with.

The solution is to use dojo.event.disconnect, which does what you might expect:

Garrick, small font below.

dojoevent.disconnect(para, "onclick", testFormContents);

dojo.event.disconnect requires that the parameters be completely identical to those used in dojo.event.connect. Once it is invoked, however, the event is disconnected.

An advanced piece of the event system is known as advice, a term that always has confused me, but which is common in the worlds of Lisp and aspect-oriented programming. The basic idea behind advice is that you can tell the system to invoke a function before or after another function. (If you have used Ruby on Rails, this is analogous to a filter.) This is admittedly an advanced feature, but it might help when debugging an application—rather than inserting logging statements into a problematic function manually, you simply can add advice to the function, invoking the logger before or after the function is invoked.

There is even a topic mechanism for Dojo events, which lets you create multiple channels for event notifications. (This is similar in some ways to the syslog facility in Linux and UNIX.) Thus, a particular object might register its interest when particular events happen on another object.

Finally, Dojo events are used to give functionality to widgets—Dojo's name for GUI elements made up of HTML and CSS.

**Ajax in Dojo**

Now that we understand how to create and use Dojo events, we can look at how to perform Ajax queries using Dojo. As you may recall, Ajax (which stands for Asynchronous JavaScript and XML) is a paradigm for Web development that uses the browser's ability to make HTTP requests behind the scenes. Combining such background HTTP requests with JavaScript, the DOM and CSS makes it possible to create more intuitive and aesthetic Web applications. We could create Ajax applications without Dojo or another toolkit, but it's much easier and more expressive to use a toolkit, if only because it means we can avoid browser differences and incompatibilities.

Listing 2 shows dojo-ajax.html, a page that contains only a single button marked “Press here”. When the button is pressed, the user sees an alert box, much as in Listing 1. But, in this version of the program, the contents of the alert box have come from a server-side program, defined in this case to be the very short hello.php (Listing 3).

**Listing 2. dojo-ajax.html**
Listing 3. hello.php9609l3.qrk

```php
<? echo "Hello from the server!"; ?>
```

The button itself is defined as we might do with any button to which we expect to attach an event, with an id attribute. It sits inside of a very small HTML form, named “theForm”:

```html
<form id="theForm">
  <input type="button" id="theButton" value="Press here" />
</form>
```

Using Dojo events, we connect the button to a function (ajaxAlert):
The only remaining question is what the ajaxAlert function does. First, it creates a JavaScript object literal and assigns it to the local variable ajaxArgs. This object literal assigns several names that will help our Ajax call work: url is the URL of the server-side program that will respond to our Ajax call, error indicates which function should be invoked if an error occurs, load indicates what function should be invoked if the call to url is successful and mimetype defines the MIME type we expect to receive from the server.

One of the annoying aspects of some other JavaScript toolkits is that they require you to create your own list of name-value pairs to be submitted in the Ajax request. This is not the case with Dojo. By setting the formNode name in our object literal to a form node, we can rest assured that all the form elements will be passed to the server. In this particular case, that is not necessarily useful or interesting, but it certainly saves some programmer time and increases program readability.

Finally, our ajaxArgs object is bound, and we're off and running. Clicking on the button means the associated Dojo event is invoked, which is ajaxAlert. That function, thanks to dojo.io.bind, then sends its arguments to the defined URL and invokes the load function upon successful completion. This is surprisingly straightforward and opens up many possible avenues for using Ajax in applications.

### Conclusion

Dojo, which we explored over the last two installments of this column, and Prototype, which we looked at in the January 2007 issue, are both strong libraries for Web developers looking to improve the quality of their JavaScript. Each has a different style associated with it. I tend to be more of a Prototype kind of guy, but many aspects of Dojo are also quite appealing to me. In particular, Dojo's extensive set of widgets, and the way those widgets can be connected to one another via the event system, provides a rich set of functionality that all JavaScript developers can enjoy. Even if you don't plan to use Dojo, you should consider installing and trying it, just to understand how it works.

### Resources

The main source for information about Dojo, as well as Dojo software releases, is [http://dojotoolkit.org](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](http://dojotoolkit.org/docs), and Dojo.book (the Wiki-based Dojo documentation) is at [http://manual.dojotoolkit.org/index.html](http://manual.dojotoolkit.org/index.html).

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

Finally, a noteworthy introduction to Dojo events is at [http://www.dojotoolkit.org/docs/dojo_event_system.html](http://www.dojotoolkit.org/docs/dojo_event_system.html).