CHAPTER 10

Shopping in Style: A Drag-and-Drop Shopping Cart

Most people who have shopped on the Internet are familiar with the shopping cart metaphor. You see a list of items for sale, you click some button, and some quantity of the item is added to your shopping cart. You then check out, and your order is processed based on the content of your cart. This is all well and good, but we can do a little better than that, can't we?

In this chapter, we will build a shopping cart that lets you drag items into it, rather than needing to click a button to add to your cart. We will use some special effects to make this look cooler than it might sound. At the same time though, we will respect the fact that some people may not have JavaScript available. For them, we will ensure our shopping cart degrades gracefully and still works, even under those "arcane" conditions.

We'll use a new library, MochiKit, for the drag-and-drop action. Also, you'll see a new technique that can come in very handy: a mock server, to handle your server needs without having to write actual server code.

So, break out your wallet and credits cards, and let's go shopping!

Shopping Cart Requirements and Goals

A shopping cart is a well-known paradigm used on most e-commerce web sites. Sites such as Amazon (`http://www.amazon.com`), Best Buy (`http://www.bestbuy.com`), Newegg (`http://www.newegg.com`), CD Now (`http://www.cdnow.com`), and TigerDirect (`http://www.tigerdirect.com`)—to name just a few—use a shopping cart to allow their users to purchase their products. It's conceptually not a difficult beast, but with a little added pizzazz, it can actually be more fun to use, and certainly a bit more Web 2.0-ish. Let's enumerate our goals for our Shopping Cart application:

- Users should be able to select an item, select a quantity of that item, and have it added to their cart. They should be able to do this by dragging items into the cart, or in a more typical manual fashion (part of graceful degradation, as you'll see shortly).
- Users should be able to view the contents of their cart at any time, including the current dollar total, and also be able to modify it from that view: add and remove items, as well as change quantities.

- Users should be able to check out; that is, complete their purchase. However, since this is a book focused on JavaScript and the client in general, we're not going to actually write that part.[1] Moreover, we're going to write a "mock" server on the client. It would be a simple exercise to replace this mock server with the real McCoy.
- The shopping cart should degrade gracefully; that is, work whether or not JavaScript is enabled. When it is enabled, the full drag-and-drop experience is available. When it is disabled, the cart still functions just fine, but in a more manual fashion, typical of most shopping carts. Because we're faking the server side though, we can't literally disable JavaScript, because we need it to do that fakery. Therefore, we'll provide a switch to turn JavaScript on and off in a fake way, so we can see how things react in either circumstance.
- We'll use a JavaScript library to help us with the more complex pieces, specifically the drag-and-drop functionality. That library is MochiKit.

With this application, we will have three primary concerns. First, we want the user to have the ability to drag items onto the cart, so we'll need to deal with the code to enable drag-and-drop. This isn't the most difficult thing to implement in a web application, but it involves a fair number of details, so we'll be using the MochiKit library to handle the complexity for us.

Second, we want the application to degrade gracefully, so it will still work when JavaScript is disabled. This book, being about JavaScript, hasn't focused on how this can be accomplished, so this will be a good chance to demonstrate how to do it. As you can imagine, a drag-and-drop shopping cart just isn't viable without JavaScript, so it truly will be a degraded experience, if one considers the drag-and-drop version to be the pinnacle. Still, we can continue to provide a perfectly usable shopping experience, even without JavaScript.

The third consideration is that we aren't going to mess with the server side of the equation here, yet we will need a server in the mix to do it right. How can we accomplish that? We can pull off this trick by utilizing a technique I like to call a *mock server*.[2]

Let's start with how we will pull off that graceful degradation.

Graceful Degradation, or Working in the Stone Age

These days, writing a web application without JavaScript is tantamount to trying to start a fire by rubbing two sticks together. There's no question it can be done that way, but why would you want to, when you can grab a BIC from Wal-Mart and do it with the flick of a finger?

In years gone by, JavaScript had a very poor reputation on a number of fronts: security, performance, annoyance (which is really a failing of those using it), and so on. In those days, people often would disable JavaScript entirely to make their browsing experience more enjoyable. Some people still do that today, even though it's far less common. Also, we should consider

1. You will be seeing some server-side code in the chapter on Ajax (Chapter 12), but that's because there isn't a particularly good way to fake it in a purely client-side manner. Here, we can do that with a mock server.
2. While I won't claim to have made up the term, I can honestly say I've never heard it referred to as such anywhere else, so, if you use it, send the royalty check to my publisher for forwarding along to me. Just kidding!