It creates a list of table rows, one for each item in the cart. Whenever you find yourself iterating like this, you might want to stop and ask yourself, is this too much logic in a template? It turns out we can abstract away the loop using partials (and, as we'll see, this also sets the stage for some AJAX magic later). To do this, we'll make use of the fact that you can pass a collection to the method that renders partial templates, and that method will automatically invoke the partial once for each item in the collection. Let's rewrite our cart view to use this feature.

Download depot_j/app/views/store/add_to_cart.rhtml

```rhtml
<div class="cart-title">Your Cart</div>
<table>
  <%= render(:partial => "cart_item", :collection => @cart.items) %>

  <tr class="total-line">
    <td colspan="2">Total</td>
    <td class="total-cell"><%= number_to_currency(@cart.total_price) %></td>
  </tr>
</table>

<%= button_to "Empty cart", :action => :empty_cart %>
```

That's a lot simpler. The render method takes the name of the partial and the collection object as parameters. The partial template itself is simply another template file (by default in the same directory as the template that invokes it). However, to keep the names of partials distinct from regular templates, Rails automatically prepends an underscore to the partial name when looking for the file. That means our partial will be stored in the file _cart_item.rhtml in the app/views/store directory.

Download depot_j/app/views/store/_cart_item.rhtml

```rhtml
<tr>
  <td><%= cart_item.quantity %>&times;</td>
  <td><%= h(cart_item.title) %></td>
  <td class="item-price"><%= number_to_currency(cart_item.price) %></td>
</tr>
```

There's something subtle going on here. Inside the partial template, we refer to the current cart item using the variable cart_item. That's because the render method in the main template arranges to set a variable with the same name as the partial template to the current item each time around the loop. The partial is called cart_item, so inside the partial we expect to have a variable called cart_item.

So now we've tidied up the cart display, but that hasn't moved it into the sidebar. To do that, let's revisit our layout. If we had a partial template that could display the cart, we could simply embed a call to

```rhtml
render(:partial => "cart")
```