```
$("cart").update("<h1>Your Cart</h1>\n\n<ul>\n  \n  <li
id=\"current_item\">\n\n  3 &times; Pragmatic Project
Automation\n</li>\n</ul>\n  \n<form method=\"post\"
 action=\"/store/empty_cart\" class=\"button-to...
```

Clearly this won't do. We need to have our application work if our users have disabled JavaScript in their browsers. That'll be our next iteration.

9.5 Iteration D5: Degrading If Javascript Is Disabled

Remember, back on page 128, we arranged for the cart to appear in the sidebar. We did this before we added a line of AJAX code to the application. If we could fall back to this behavior when JavaScript is disabled in the browser, then the application would work for Bruce as well as for our other co-workers. This basically means that if the incoming request to add_to_cart doesn't come from JavaScript, we want to do what the original application did and redirect to the index page. When the index displays, the updated cart will appear in the sidebar.

If a user clicks the button inside a form_remote_tag, one of two things happens. If JavaScript is disabled, the target action in the application is invoked using a regular HTTP POST request—it acts just like a regular form. If, however, JavaScript is enabled, it overrides this conventional POST and instead uses a JavaScript object to establish a back channel with the server. This object is an instance of class XmlHTTPRequest. Because that's a mouthful, most folks (and Rails) abbreviate it to xhr.

So, on the server, we can tell that we're talking to a JavaScript-enabled browser by testing to see whether the incoming request was generated by an xhr object. And the Rails request object, available inside controllers and views, makes it easy to test for this condition: it provides an xhr? method. As a result, making our application work regardless of whether JavaScript is enabled takes just a single line of code in the add_to_cart action.

Download depot_o/app/controllers/store_controller.rb

```
def add_to_cart
  begin
    product = Product.find(params[:id])
  rescue ActiveRecord::RecordNotFound
    logger.error("Attempt to access invalid product #{params[:id]}")
    redirect_to_index("Invalid product")
  else
    @cart = find_cart
    @current_item = @cart.add_product(product)
►   redirect_to_index unless request.xhr?
  end
end
```