Prototype: Ajax

1. Make a JSP page that outputs a random number. Use Ajax.Request to make a push-button that, when pressed, gets the random number and pops it up in an alert box.

2. Make a button that, when pressed, gets a random number from the server and inserts it into the page.

3. Make a form with three textfields (person1, person2, person3). Send the values to the server, and have the server send back a bulleted list (i.e., `<ul>` list) showing the names.

4. Make a form that collects a banking customer id and then, when a button is pressed, shows a list of the id, first name, last name, and balance. Give an error message for unknown ids. Use JSON on the server.

You have done almost-identical tasks twice before, but use Prototype this time. If you never completed any of the previous Customer exercises, then just use a JSP page that returns a JSON object with three numbers in it:

```
{ num1: <%= Math.random() %>,
  num2: <%= Math.random() %>,
  num3: <%= Math.random() %>
}
```

Then, just do something simple with the result. The point is to get experience with Ajax.Request and the use of the responseJSON property.

5. Improve the previous example in two ways:
   - Show some sort of “working...” message while you are waiting for the response, and make the server response slow enough that you can see this in action.
   - Don’t hardcode the name of the result region in the JavaScript function, but instead pass it from the HTML page (not from the end user: from the HTML source code). You might think this is hard since the function you pass to onSuccess takes one argument only, but think about the way we have done response handlers in the past, and this should be very easy.

6. Make a class called AjaxRequest that works like Prototype’s Ajax.Request, with the exact same syntax. Don’t use any Prototype Ajax.Xxx or Element capabilities to do so. For the options, you only need to support the onSuccess and properties properties, and for properties, you only have to support an explicit parameter string (not a parameter object as in the later class examples). Note that I am not asking you to implement the responseJSON property or the “$”, “$F”, and “output” methods: only the core functionality of Ajax.Request.

Hint: this is just a few lines of code, and you don’t need Class.create or any other fancy OOP features. Just a constructor function.