Exercises: jQuery Selectors

1. Make a page that has a few bulleted lists in it, and loads jquery.js. Deploy the app, go to the page, then bring up FireBug. Now, try a few things interactively:

   • Find all the li elements
   • Find all the li elements and make them disappear. Then make them reappear.
   • Make li elements disappear only if they have a css class name of “test”
   • Make li elements turn blue. Revert them to their original state.

2. Make a button that, when pressed, turns all li elements blue. Make another button that returns them to their original state. Use explicit onclick handlers in the HTML.

3. Repeat the above problem, but avoid any explicit JavaScript in the HTML page (i.e., use the “unobtrusive JavaScript” approach that assigns to the click handler programmatically after the DOM is loaded).

4. Make a button that, when pressed, turns li elements blue only if the list is inside a whose CSS class is colorChangeable. Make another button that returns them to their original state. Make sure your HTML page has at least one <ul> inside such a div and at least one <ul> outside. For example, here is some HTML that has li elements that match:

```
<div class="colorChangeable">
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
</div>
```

5. Make a button that, when pressed, turns li elements blue only if the list is inside a whose CSS class is colorChangeable, *and* only if the li element has a class called modifiable. Make another button that returns them to their original state.

6. Make a button that, when pressed, slowly hides all li elements. Make another button that makes them reappear.

7. **[Harder]** Make a button that, when pressed, fires off an Ajax request. The server should return the name of an HTML element or CSS selector, and your client-side code should hide all of elements that match. Have the server use Math.random() to return different HTML element names (or CSS selector strings). Hit the Reload button in the browser to return it to its original state. If you are unfamiliar with Java servlets, modify the exercise by making an Ajax request that returns the contents of a text file called element-name.txt, and put a fixed element name inside that text file. Even though this is not quite as much fun since the results are the same each time, your client-side code would be *exactly* the same as with the version where the Ajax request randomizes the content.