6.813/6.831 • User Interface Design and Implementation

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
Spring Semester, 2011

PS1: Implementing an HTML/Javascript Application

Due Session 11

This problem set is for the undergraduate 6.813 class only.

This problem set asks you to build a small user interface that searches a list of words using the set of core technologies that drive most modern web applications, namely HTML, CSS, and Javascript. We provide some backend code that does the heavy lifting (actually loading the word list and searching it). We also specify the design of the user interface. Your job is to implement it using HTML, CSS, and Javascript.

To do this assignment, you'll need to know how to:

- write HTML: create correct HTML pages that display in modern browsers
- use HTML form elements (like <button>, <input>, and <select>) to create a web application
- use CSS to layout and style HTML elements
- use Javascript (and jQuery) to add event handlers and bindings to respond to user input;

Here are some useful reference sources for HTML/CSS/Javascript:

- HTML Dog (a fairly good guide for both HTML and CSS)
- W3Schools (a reference for learning HTML, CSS, and Javascript)
- CSS Tutorial (a tutorial for learning CSS)
- How jQuery Works (a beginning tutorial on jQuery, the Javascript library we are using for this assignment)
- jQuery Documentation (the official documentation for jQuery is an excellent resource)
- O'Reilly Safari has several e-books on HTML, CSS, and Javascript.
  Dynamic HTML: The Definitive Reference, 3rd Edition is a good one to start with.

Provided Resources

We provide you with the following:

- ps1.zip: a zip file containing the code you will be editing for this problem set.

You can import this zip file directly into Eclipse using File/Import/Existing Projects into Workspace, or use whatever text editor you would like. In the root folder of the project are the following files:

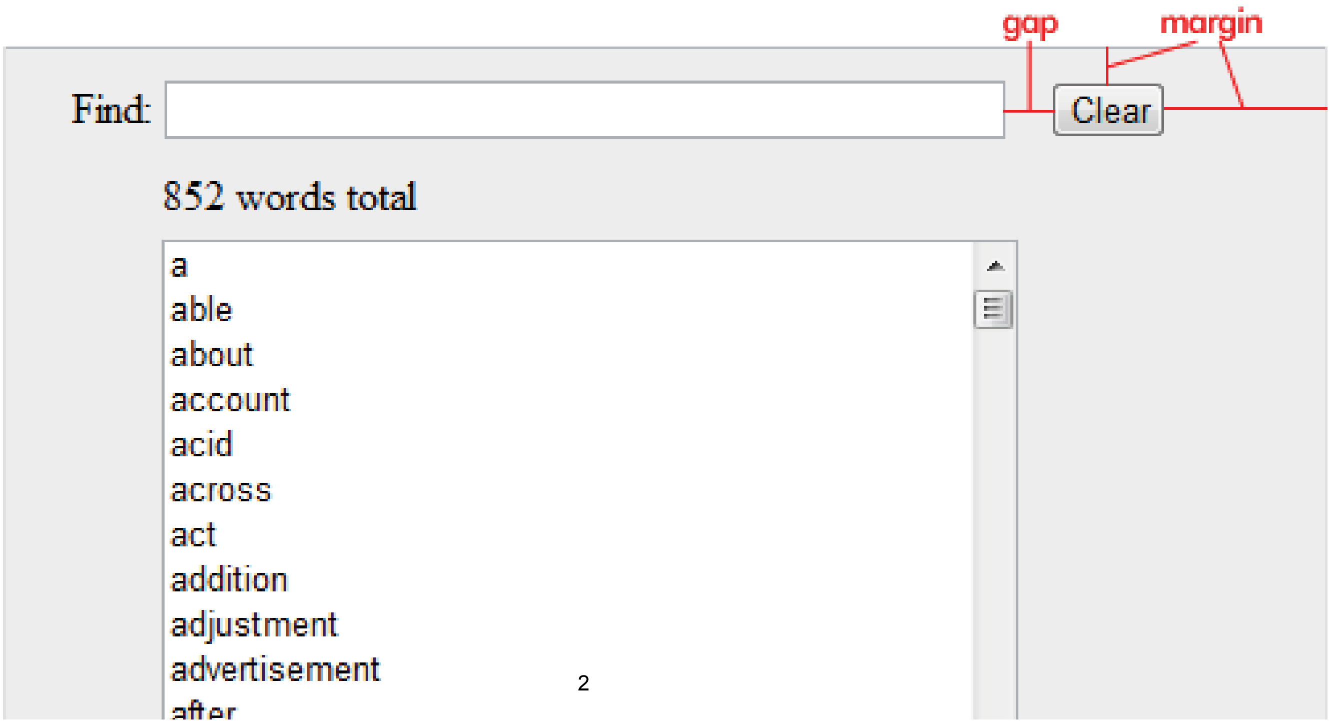
- jquery-1.5.min.js: the code for jQuery, the Javascript library you will be using for this assignment.
- words-small.json: a dictionary of 850 common English words in JSON notation.
- words-large.json: a dictionary of 45,407 words taken from the standard Linux /usr/share/dict/words, in JSON notation.
- dictionary.js: a backend class that represents a list of words and provides operations for filtering the list using a search pattern.
- word-finder.html: a skeleton file for your user interface.

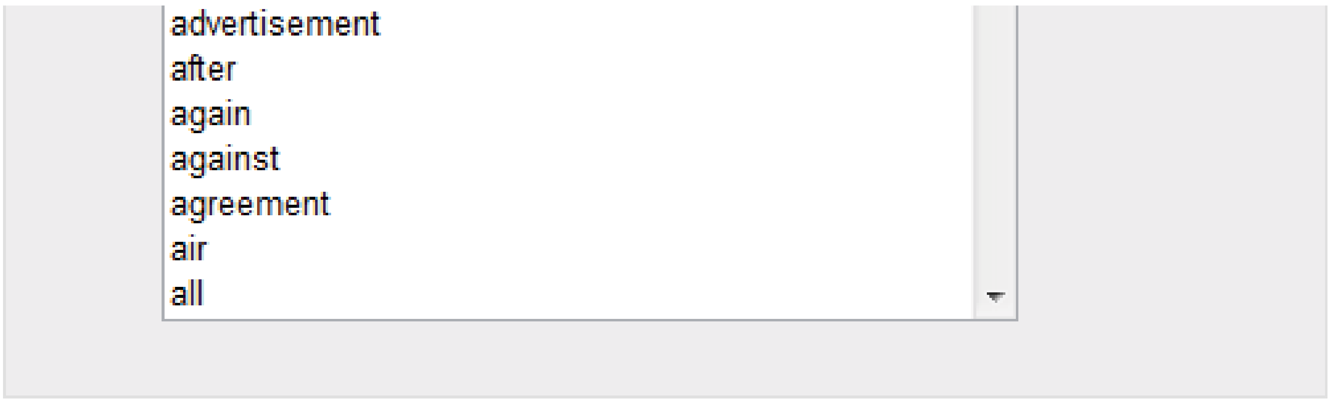
Feel free to change these files as you see fit.

If you are using Eclipse to edit these files, we highly recommend installing the appropriate plugins for web development support. On the latest release of Eclipse (Helios), you can do this by going to Help/Install New Software..., then selecting the Helios update site, then selecting Web, XML, and Java EE Development/Eclipse Web Developer Tools and Javascript Development Tools.

Basic Layout (30%)

First, build an interface that looks like this. There should be margins between and around the elements as shown. No



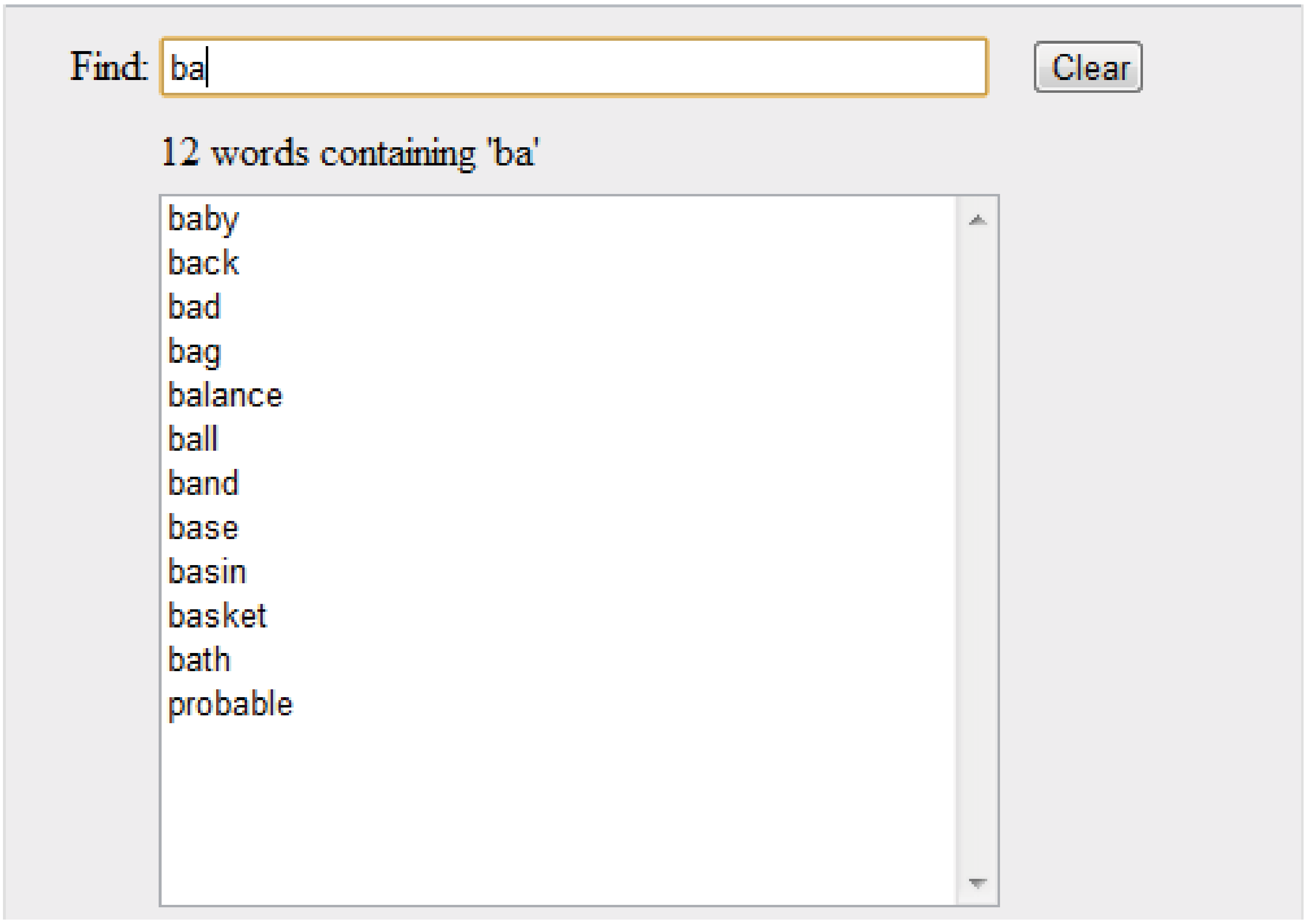


behavior is required to earn the points for this part, just a static, fixed-size layout. The <select> widget (which is showing words in the figure) should have a handful of words in it. You will want to use HTML form elements whenever appropriate.

Displaying Words (20%)

Improve your interface so that when the program is first run, the select element displays the small dictionary we gave you, as shown in the figure. The label above the select element should display the number of words in the word list, as shown.

Searching (40%)

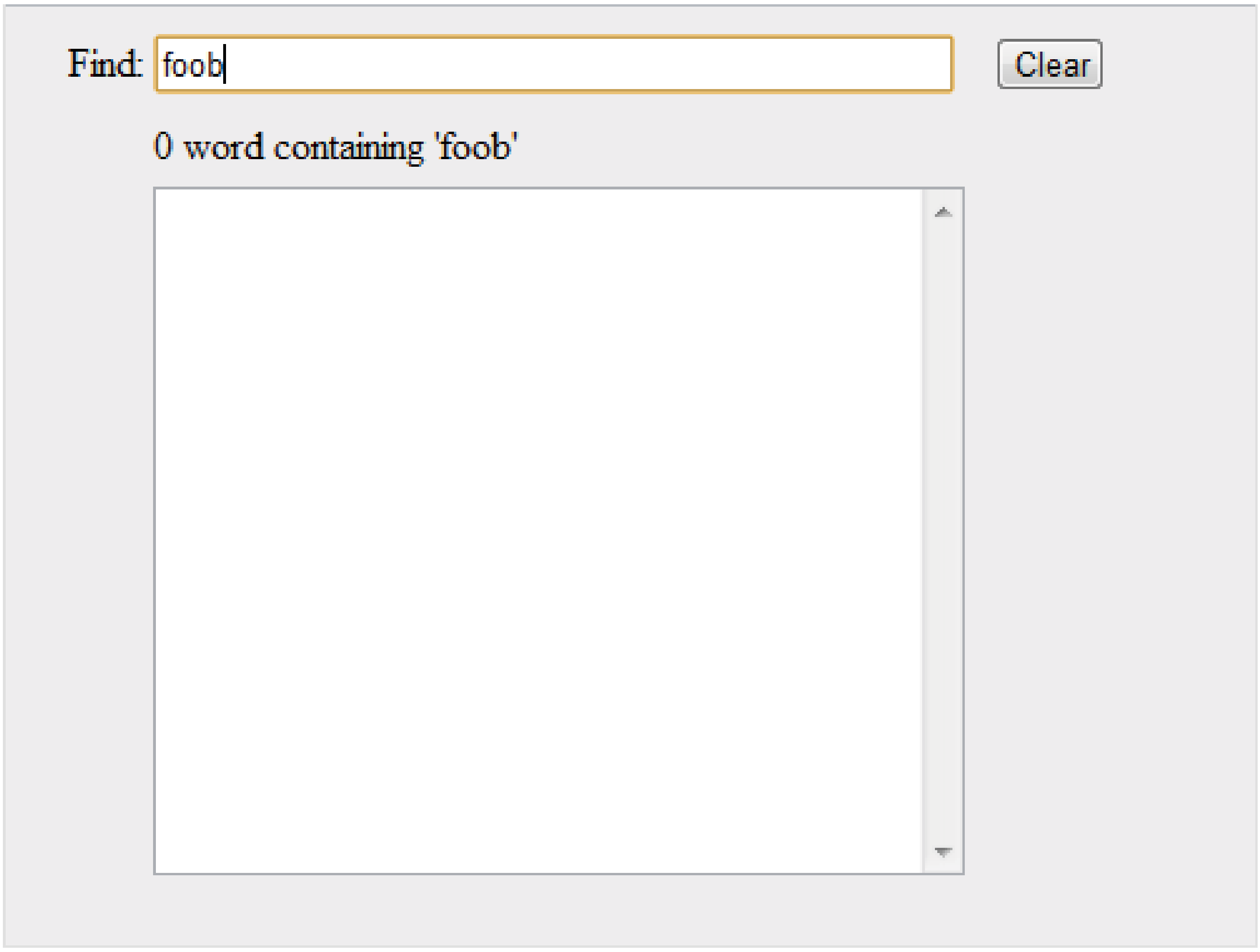


The Find input field contains the user's query. When the query is blank, the list box displays the entire word list, as shown above. Whenever the query changes, the list box

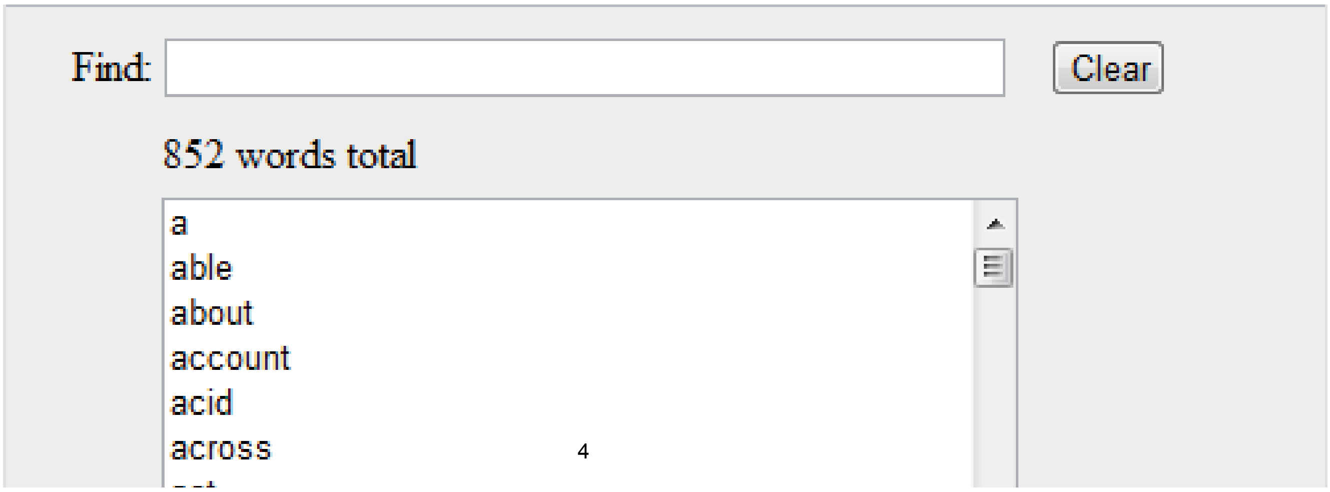
immediately updates to display all words that contain the query text.

The list box should update constantly as the user types. Pressing Enter should not be necessary. (Hint: jQuery's event handling API makes this particularly simple, see the documentation here.) Be careful about which keyboard-related events you decide to listen to.

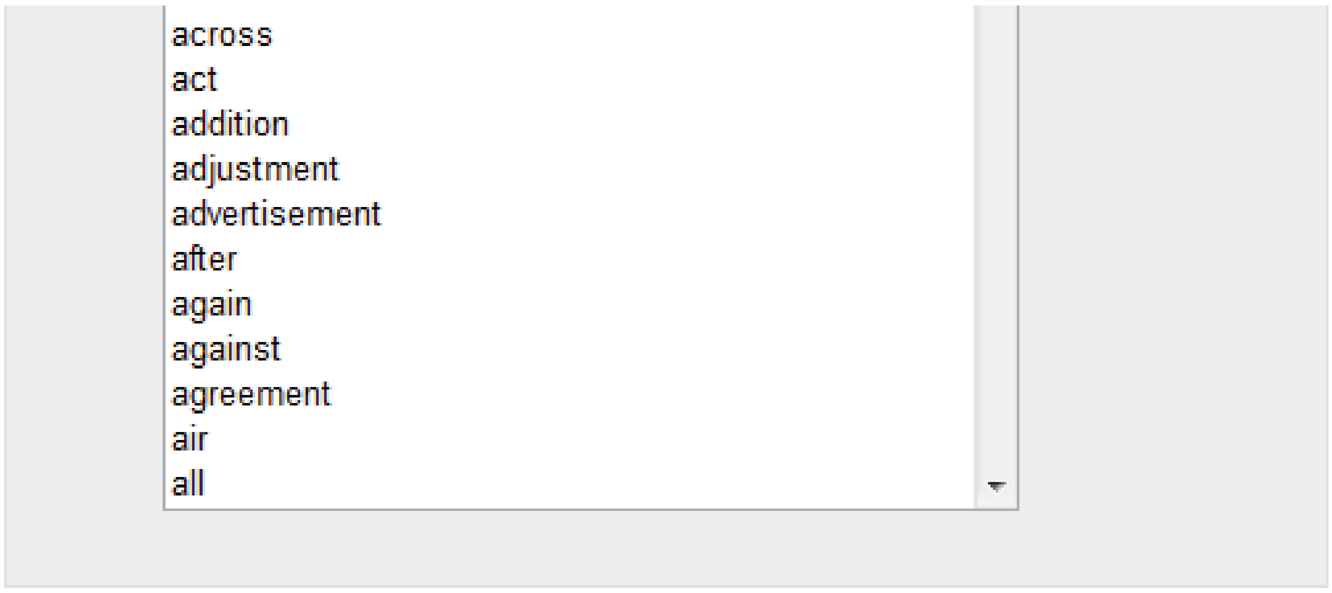
If none of the words contain the query, the list box should be empty.



The Clear button should clear the query field, restoring the list box to

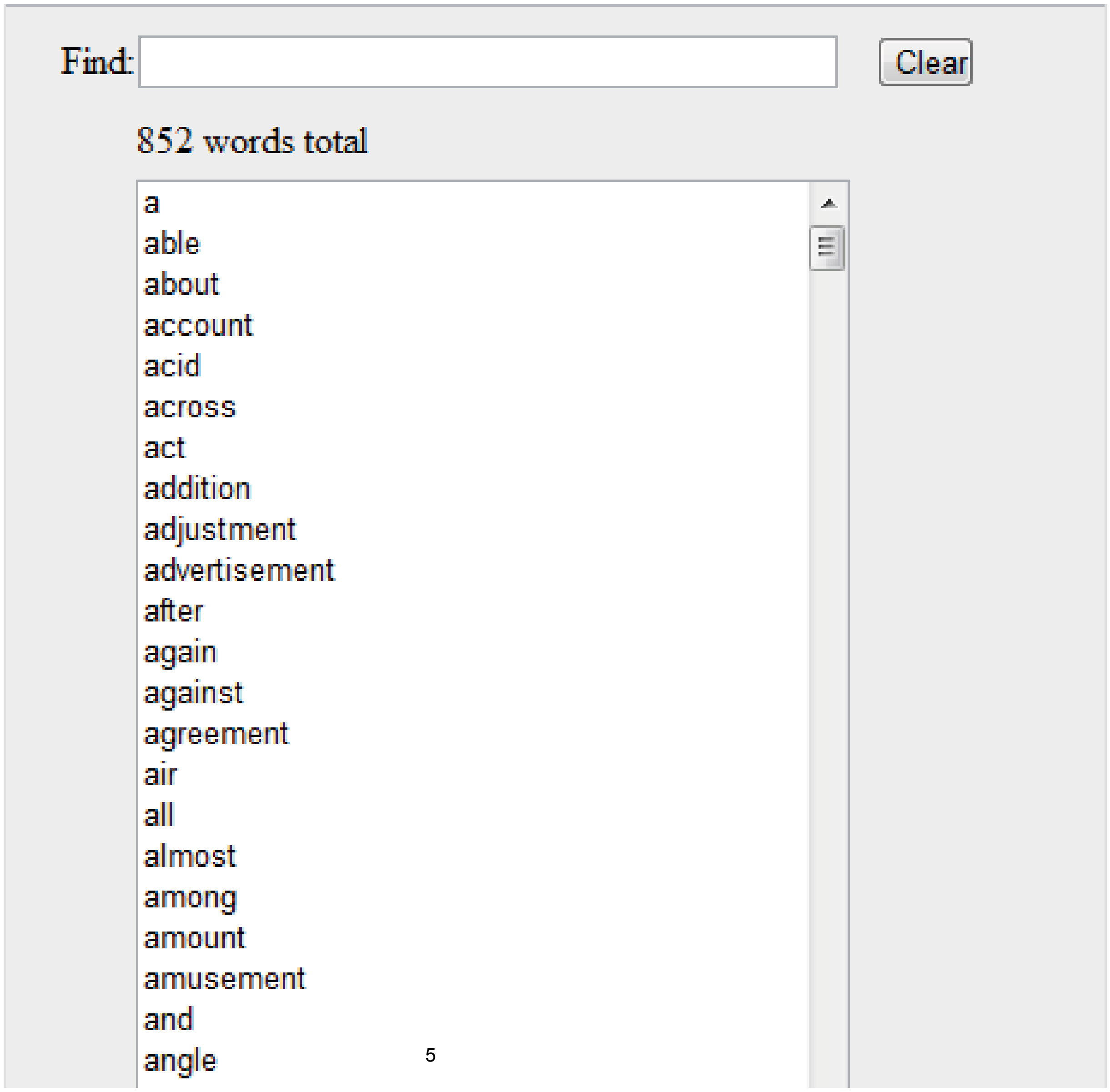


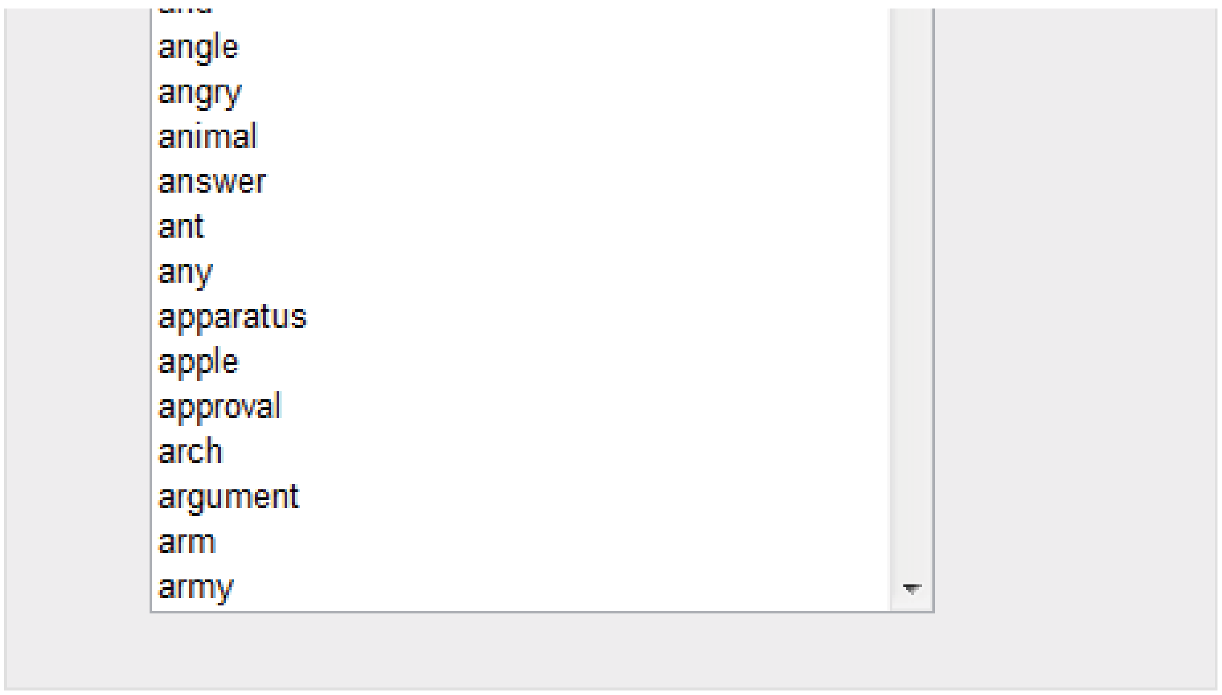
displaying all words again.



Resizing (10%)

The window should be resizable, with all extra space going to the query field and the word list. You may need to use some Javascript to dynamically set the height of the <select> word list based on its parent's height.




angle
angry
animal
answer
ant
any
apparatus
apple
approval
arch
argument
arm
army

Further (Optional) Improvements

If you found this assignment easy and you're inclined to go further, here are some ideas for optional improvements:

- Support for the large dictionary of words (about 45,000) without compromising the responsiveness of the application. This requires intelligently lazy-loading search results into the HTML DOM only as the user scrolls to them.
- Dynamically query an external web service for search results, most likely as an AJAX request. This would allow us to use an almost arbitrarily large dictionary, since we can delegate searching to the server.
- Highlight the matching substring of each word in the list box, so that the user can see at a glance how the word matches the query.
- Extend the interface with new controls and new behavior so that the user's query can be a crossword puzzle entry with some letters filled in and some letters empty.

What to Hand In

Package your completed assignment as a zip file that contains all of your HTML, JS, and CSS files (including the JSON data).

List your collaborators in the comment at the top of word-finder.html. Collaborators are any people you discussed this assignment with. This is an individual assignment, so be aware of the course's collaboration policy.

Here's a checklist of things you should confirm before you hand in:

1. Make a fresh folder and unpack your zip file into it
2. Make sure your collaborators are named in word-finder.html
3. Make sure all assets used by your code are found in the fresh folder and load successfully
4. Make sure that the page renders correctly in at least two modern, standards-compliant web browsers (Firefox, Chrome, Safari)