### SER421 ONLINE: Web Applications and Mobile Systems Fall 2016

### Lab 5, due Thursday, 11/17/16 at 11:59:00pm via online submission to Blackboard

The goal of Lab 5 is to get you working in a browser-based application, using various techniques such as DOM manipulation, event handling, and storage. Submission instructions are at the end, PLEASE FOLLOW SUBMISSION INSTRUCTIONS!

**Activity 1: DOM expressions (20 points)**

Go to [www.bing.com](http://www.bing.com) and perform a search using 3 distinct words of your choosing. Save the resulting page (save the complete page, not just the "source"). Load the page you just saved locally (Open File…<file you just saved>).

Now, for this activity, write DOM expressions that do the following:

1. (3) Output to the console the <ol> element encompassing the results of the search
2. (4) Output to the console the code for the "onload" event on the <body> element
3. (3) Output to the console the 2nd child node underneath the <body> element
4. (3) Output to the console the number of <h2> tags in the page
5. (3) Output to the console the value in the search bar (you must get this from the search bar not anywhere else on the page)
6. (4) Make the "Images of" result near the top of the search result go away

**Activity 2: Implement your own Eliza ON THE CLIENT! (60 points)**

For Lab 2 you implemented an Eliza on the server – now we will port Eliza to the client (browser)! To do this we will make some simplifying assumptions compared to the server-side Eliza you already created.

First, you will use the dictionary structure provided to you on theBlackboard site. This dictionary format was already posted by your classmates so it is hopefully familiar to you. Requirements:

Create your own Eliza under the following constraints:

1. Your application has NO server component whatsoever.
2. Your application has NO CSS.
3. Your application must be a "single page application" – that is, it never reloads a page from a local source or does a document.write() to simulate a page refresh. The page is in effect your desktop application GUI

Functionally your program must:

1. (5) Greet the user by asking for her/his name on startup in a simple form. The name should be "remembered" and used anywhere a direct naming of the end user is appropriate. Eliza should then start the conversation with a question. The question is not fixed, but should start with common greetings like "<name> how is your day going?" "<name>, is something troubling you?" or "<name> you seem happy, why is that?"
2. (15) Provide a one-line web form that allows the user to "talk" to Eliza. You should parse the string the user types in and search for matches in the dictionary and select one for a response. Echo the user's input and Eliza's response "above" the web form (that is, the one-line web form should always be at the bottom of the page). *For echoing the input and Eliza's response, you MUST manipulate the DOM of an HTML element, not a textarea!*
3. (15) You should vary the responses to the same keywords (although the number of responses will of course be finite). Vary responses by
   1. Remembering responses to the same keyword in that session
   2. Introducing some simple randomization so no 2 sessions follow the same pattern of responses. For randomization, use the basic Math.random() built into Javascript.
4. (10) If the user does not respond to an Eliza question within 20 seconds, Eliza should display a dialog box with a message such as "<name>, I'm waiting here!" or "Whatsa matter <name>, cat got your tongue?" or so forth (have some fun with it, but it must include <name>, where <name> is the end user's name). Again the prompt should not always be the same. To implement this feature, look up the window.setTimeout API in the browser.
5. (10) Your program should detect the presence of JSON input *into the user input form*, and have the ability to dynamically incorporate the new JSON into dictionary by adding (not replacing) its entries to the existing dictionary Eliza is using. When this happens, Eliza should proudly announce: "I just got smarter!" (Note in your server-side implementation we did this with the fs module, here we will do it through user input).
6. (5) Provide a "clear" special operation that returns the user back to the starting point (back to step 1)

**Activity 3: Make Eliza Stateful (20 points)**

Extend your activity 2 solution with 2 new features:

1. (12) Make your Eliza program stateful by saving the responses it gives to <name>. If the browser closes and restarts, and you come back to Eliza, and enter the same <name> as a prior respondent, then you should be able to restore the state of the application. That is, modify requirement 3 from activity 2 so that your app restores its "session" for that user.
2. (8) Modify the special "clear" operation so that it clears the state of the application for <name>, then returns the app to the start form.

**Submission Instructions:**

Submit your lab as a single zipfile named <asurite1>\_ser421lab5.zip with the following structure:

1. In a subdirectory "activity1", have the complete web page that is the result of your Bing search. Have a javascript file named activity1.js that has the sequence of expressions used to answer activity 1. Please put comments before each block of code to label it with each step of activity 1. Then save the source of your modified file (after running your activity1.js) as activity1.html (this is not the web page "complete", it is just what you get when you "view source". You can cut-and-paste into an editor and save).
2. THIS LAB MUST BE DONE INDIVIDUALLY.
3. Save your answer to activity 2 in the root directory under the name activity2.js
4. Save your answer to activity 3 in the root directory under the name activity3.js
5. As always if there is anything you want us to know, put a README.txt in the root directory.
6. And again, you can submit a partial credit if not done in activity 2 or 3. Name it activity2[3]pc.js and explain what we should expect to find in the README.txt. We allow unlimited submissions so there is no reason to be late!