CSCE 120: Learning To Code

Consuming Data I
Hacktivity 11.2

Introduction

Prior to engaging in this hacktivity, you should have completed all of the pre-class activities as outlined in this module. At the start of class, you will be randomly assigned a partner to work with on the entirety of this hacktivity as a *peer programming* activity. Your instructor will inform you of your partner for this class.

One of you will be the driver and the other will take on the navigator role. Recall that a driver is in charge of the keyboard and computer while the navigator is in charge of the handout and directing the activity. However, you are *both* responsible for contributing and discussing solutions. If you were a driver/navigator in the prior activity, switch roles for this activity.

You will use the same project from the previous Hacktivity. If you need to, you can re-download it from GitHub using the URL, `https://github.com/cbourke/BulletinApp`.

1 Analyzing an Ajax App

Open the files under the `course` folder. Evaluate the HTML file and experiment with it: try looking up the courses you are currently taking this semester. Now try looking up a course that does not exist.

Examine the HTML and JavaScript files and answer the following questions:

1. What URL does the Ajax call connect to?
2. What parameters are submitted in the request?
3. Try making the same request in your favorite browser and examine the raw JSON data

4. What functions are responsible for the display of the course information (upon success) and error display messages and how do they work?

2 Writing an Ajax App

In this activity, you'll write an Ajax App that allows a student to search UNL's course bulletin. We have provided some starter files in the `bulletinSearch` folder. The HTML file contains a simple "search" bar that allows a user to enter search terms. We'll connect to a course search webservice that is supported by UNL's course bulletin.

The webservice can be accessed using the following URL (which has set a CORS policy so that we can connect to it from any origin):

https://bulletin.unl.edu/undergraduate/courses/search

The API accepts several parameters:

- `q` – a search term (or terms) to be queried (as a single string)
- `format` – the format of the response (we'll want `"json"`)
- `limit` – a numerical value indicating the maximum number of courses to be included in the response

The function, `searchBulletin()` has been provided, but is empty. You will need to complete it by:

1. Retrieving the search query string from the search bar
2. Formulating and executing a proper `$.ajax()` call
3. Writing both `success` and `error` functions to process the search results.

Be sure to:

- Handle errors appropriately (display an error message)
- Handle situations in which there are no results
- Give appropriate visual cues to give the user the best UX

To process the results, we suggest that you view them in a regular browser. Cut and paste the JSON to a online JSON formatter tool like http://jsonformatter.curiousconcept.com/.

3 Using Ajax in Autocomplete

This is an *optional* advanced activity for those wanting to learn a bit more about using ajax and jQuery UI.

Let's make the search experience more dynamic by using the autocomplete feature of jQuery UI. This feature enables functionality in a text box to "auto suggest" possible results to the user so that they do not have to enter a full search term.

UNL's bulletin does something similar, observe the results of entering a partial search term (say "Learning") in the search bar of the following page: http://bulletin.unl.edu/undergraduate/.

Recreate this functionality using jQuery UI's autocomplete widget. Full documentation can be found here:

https://jqueryui.com/autocomplete/

In particular using autocomplete with a remote data source example:

https://jqueryui.com/autocomplete/#remote

specifically, you will want to use a function as a source in which you make an ajax query:

http://api.jqueryui.com/autocomplete/#option-source