Module 3 Coding Assignment

Click here to SIGN UP for the ***Single Page Web Applications with AngularJS*** course on Coursera. It's FREE!

Time to put all that knowledge to code!

Time to Complete

It should take about an 1 hour or less. (This is just an estimate. People's backgrounds differ, so for some people it will take longer to complete.)

Ask questions in Week 3 Discussion Forum if you get stuck! We are all learning, and going through getting stuck and then unstuck (even with someone's help) can be a very valuable learning experience!

DO NOT be scared by the length of this assignment! It's actually a fairly short one. I just wanted to explain everything as clearly as I could and break it down into smaller steps for your benefit.

Assignment Instructions

General Idea

You are going to be building a much simplified search of the menu item descriptions using the restaurant server REST API we used in Lecture 25, Part 2.

The idea here is for the user to search the descriptions of menu items. Then, given the list of matches of his search, give the user the ability to throw the items they for sure don't want off the list, thus narrowing it down to what they do want.

Your task is create a text box and a button with the label "Narrow It Down For Me!".

Initially, the user should just see a screen with the textbox and the "Narrow It Down For Me!" button. Once the user enters something into the textbox and clicks the button, your app will reach out to the server and retrieve the list of menu items for the entire menu. Once retrieved, your task is to loop through all the items and, for each item, do a simple check if the string being searched for by the user appears anywhere in the description of the item. If it does, that item gets placed in a special `found` array. If it doesn't, you simply move on to the next item.

Once your app goes through all the items, it should display the `found` list of items. Each item in the list should show the name of the menu item, its short_name, and the description. You can display the

items in a simple unordered list, with each piece of information separated by a comma. OR be fancier and use some sort of a grid. Either way is fine. We are not concentrating on style in this class.

You should also provide a "Don't want this one!" button next to each item in the list to give the user the ability to remove an item from that list.

If nothing is found as a result of the search OR if the user leaves the textbox empty and clicks the "Narrow It Down For Me!" button, you should simply display the message "Nothing found".

To make things a bit easier, you can retrieve the items from the server every time the user clicks the "Narrow It Down For Me!" button. You don't have to cache the response. In other words, no matter what actions the user has taken, if the "Narrow It Down For Me!" button is clicked, all the data gets wiped out and the whole process starts all over again. No requirement to remember what the user chose to throw off the list last time.

Rules

Breaking one of these rules will cause you to fail the assignment:

- You are not allowed to use regular HTML `onclick` attribute to bind behavior to the button. You **must** use the AngularJS way of binding behavior.
- At no point should your Javascript code look up *anything* in the DOM of the browser.

Steps

Here is what you will need to do to complete the assignment:

Steps for Setups (similar to all other assignments)

1. (If you haven't already) Create a GitHub.com account and a repository that you will use for this class.
2. (If you haven't already) Follow the Development Setup Video (beginning of Module 1) instructions on how to create a repository and set it up such that you can host and view your finished web pages on GitHub Pages, i.e., GitHub.io domain name. You will need to provide that URL for your peer review.
3. Create a folder in your repository that will serve as a container folder for your solution to this assignment. You can call it whatever you want. For example, `module3-solution` or `mod3_solution`, etc.
    - You can do this by 'cloning' your repository with `git clone https://www.github.com/your_repo_url` to your local machine, creating `module3-solution` folder in the root of the repository directory along with a README.txt inside of the `module3-solution` directory. Then, you would do `git add .`, followed by `git commit -m 'New folder'`, followed by `git push` to upload the new folder with the README.txt to the GitHub repository.
4. HTML/CSS for the assignment

- Option 1: Copy the **contents** of the folder `assignment3-starter-code` into the newly created folder from the previous step. If you cloned this repository, the assignment 3 folder is located in `root_dir_of_your_local_repo/assignments/assignment3/assignment3-starter-code`
- Option 2: Create the HTML/CSS yourself. Make sure to name the HTML file `index.html`. The only requirement is that your HTML have the required parts as described in the General Idea section. The rest is up to you.

5. Import AngularJS into your project and place a `<script>` tag right before the `</body>` tag.

Steps for Implementing Assignment Requirements

1. Declare `ng-app` either on the `html` or the `body` element. Name your app `NarrowItDownApp`.
2. Create `app.js` in your project and declare an Angular module to match your `ng-app` declaration.
3. Declare and create a NarrowItDownController (with controller as syntax) that will wrap your search textbox and button as well as the list of found items.

4. Declare and create MenuSearchService. The service should have the following method: getMatchedMenuItems(searchTerm). That method will be responsible for reaching out to the server (using the $http service) to retrieve the list of all the menu items. Once it gets all the menu items, it should loop through them to pick out the ones whose description matches the searchTerm. Once a list of found items is compiled, it should return that list (wrapped in a promise). Remember that the `then` function itself returns a promise. Your method would roughly look like this:

```
return $http(...).then(function (result) {
    // process result and only keep items that match
    var foundItems...

    // return processed items
    return foundItems;
});
```

   The URL for the REST Endpoint is https://davids-restaurant.herokuapp.com/menu_items.json.

5. The NarrowItDownController should be injected with the MenuSearchService. The controller should call the getMatchedMenuItems method when appropriate and store the result in a property called `found` attached to the controller instance.
6. Declare and create `foundItems` directive. The list should be displayed using this `directive` which takes the `found` array of items specified on it as an attribute (think one-way binding with '<'). To implement the functionality of the "Don't want this one!" button, the directive should also provide an on-remove attribute that will use function reference binding to invoke the parent controller removal an item from the `found` array based on an index into the `found` array. The index should be passed in from the directive to the controller. (Note that we implemented almost identical type of behavior in the Lecture 30 Part 2, so as long as you understood that code, it should be very close to copy/paste). In the NarrowItDownController, simply remove that item from the `found` array. You

can do that using the Array's splice() method. For example, to remove an item with the index of 3 from the `found` array, you would call `found.splice(3, 1);`.

Important Implementation Notes

1. Make sure all of your Javascript code is inside of an IIFE. (*If you don't know what that is or why we'd want to use it, brush up on it by looking through module 4 of HTML, CSS, and Javascript for Web Developers course I teach.*)
2. Make sure all of your dependency injections are protected from minification.
3. After you are done and satisfied with your solution, don't forget to add/commit/push your code to your repository.

IMPORTANT REMINDERS:

- Closely follow the submission guidelines for this assignment on Coursera.org
- Make **sure** you provide the correct URL in your submission (it should be GitHub**.io**, *not* GitHub**.com**)
- Make **sure** to TEST your assignment not just on your local machine, but ALSO once you deploy it on GitHub, using the URL you are providing as part of your submission.
- This assignment will be peer-reviewed (and graded). The guidance will be given such that if submission instructions are not followed, the assignment is to be failed. This includes providing the wrong URL for your deployment. Following instructions is very much part of software development. After all, that's what software requirements are - instructions to follow.