Module 2 Coding Assignment

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

Ready to implement some cool dynamic behavior right in your web page?! Oh, and do it in an architecturally correct/elegant way?!

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 2 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 really not so much at all. 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

The idea of this assignment is to create a "check off" shopping List application.

Think of being in a store with a shopping list that allows you to "check off" the items you've already bought, except instead of checking them off, the bought item simply moves to the "Already Bought" list.

Your HTML page should display 2 lists, one titled "To Buy" and the other "Already Bought".

The "To Buy" list should be pre-populated with a list of at least 5 items. (*Hint: Use an array of object literals, where each item will be similar to* `{ name: "cookies", quantity: 10 }`) Each shopping list item is to have a name and quantity. It should be displayed to the user in the format of `Buy item_quantity item_name`. For example, shopping list item `{ name: "cookies", quantity: 10 }` would be listed as `Buy 10 cookies`.

Next to each item in the list should be a button with the label "Bought". When the user clicks on the "Bought" button, its associated item should be removed from the "To Buy" list and appear in the "Already Bought" list.

The "Already Bought" list should initially be empty and display a message "Nothing bought yet". Make sure the message appears *only* when the list is empty. Once something is "bought" and appears on this list, the format of each item in the list should be `Bought item_quantity item_name`. For example, the bought item of 10 cookies mentioned before would appear in this list as `Bought 10 cookies`.

Once the user "buys" every item on the "To Buy" list, i.e., clicks on the "Bought" button for every item in the "To Buy" list, instead of the empty "To Buy" list, display the message "Everything is bought!"

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.
- Your implementation should *not* consist of only 1 controller that does it all: take care of both lists, store the data for both lists, etc.

Steps

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

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, `module2-solution` or `mod2_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 `module2-solution` folder in the root of the repository directory along with a README.txt inside of the `module2-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 `assignment2-starter-code` into the newly created folder from the previous step. If you cloned this repository, the assignment 2 folder is located in `root_dir_of_your_local_repo/assignments/assignment2/assignment2-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 lists as described in the General Idea section. You can make the lists side by side or one under the other. The rest is up to you.
5. Import AngularJS into your project and place a `<script>` tag right before the `</body>` tag.

6. Declare `ng-app` either on the `html` or the `body` element. Name your app `ShoppingListCheckOff`.
7. Create `app.js` in your project and declare an Angular module to match your `ng-app` declaration.
8. Go back to `index.html` and declare 2 controllers using `controller as` syntax. One controller should be called `ToBuyController` and the other called `AlreadyBoughtController`. You are *required* to have 2 controllers for this assignment.
9. You will obviously need to share data between these controllers. Go back to `app.js` and implement this data sharing using the *singleton* approach with the `.service` declaration. Call the service `ShoppingListCheckOffService`. Make sure to inject this service into both controllers so they can share data. Also, realize that your service will have to keep track of both 'to buy' and 'bought' items at the same time. (*While there is no one right way to accomplish this functionality, for this assignment, you are required to implement it as described.*)
    - (*Hint*) You can store 2 separate arrays in the service: one to hold "to buy" items and one to hold "bought" items. The reference to the "to buy" array should be placed/exposed onto the `ToBuyController` instance as some property. The reference to the "bought" items array should be placed/exposed onto the `AlreadyBoughtController` instance as some property.
    - (*Hint*) When the user clicks on the "Bought" button, simply pass the call from your (`ng-click`) controller-bound method to call the right method inside of your `ShoppingListCheckOffService` service, which removes that item from the "to buy" array and pushes it to the "bought" array.
    - (*Hint*) Your `ShoppingListCheckOffService` would also be the place where you would store the initial array of "to buy" items.
10. To display and/or hide the messages when the list(s) are empty, use the `ng-if` directive.
11. To loop over the items in either list use the `ng-repeat` directive.
12. 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.*)
13. Make sure all of your dependency injections are protected from minification.
14. 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.