Entering To-Do Items

an hour

Create And Serve A To-Do Item Form

To display the form that lets you enter new items, it has to create a dropdown that contains the categories that are in the application. This is identical in *intent* to the code that creates a list of categories to display on the category screen. Because of that, here's the code that handles displaying the "new item" screen.

```
else if (req.url === "/items/new" && req.method === 'GET') {
  const filePath = path.join(__dirname, 'todo-form-screen.html');
  const template = await fs.promises.readFile(filePath, 'utf-8');
  const html = mergeCategories(template, categories, 'option');
  res.setHeader('Content-Type', 'text/html');
  res.writeHead(200);
  res.write(html);
}
```

In this case, the `mergeCategories` method is now called with the third argument of "option" rather than "li" as it was before. This is what the last three tests in the **merge-categories-spec.js** file address. You will write tests that generate "option" tags rather than "li" tags. You'll also test that the replacement correctly occurred.

In this case, you'll modify the tests in the second sub-"describe" section, the one that reads "For selects".

The first test

The first test reads

```
it("should return no <option>s for no categories", () => {
  expect.fail('please write this test');
});
```

Replace the `expect.fail` line with a test that properly follows the *Three As* of unit testing.

In the *arrange* section, you will need to create an empty array for the

`categories` and store it in a variable. You will use the variable in the action.

In the *act* section, you will invoke the `mergeCategories` function with the `template` as the first argument, the variable that contains an empty array as the second argument, and the string 'option' for the tag name as the third argument. Store the return value in a variable.

In the *assert* section, assert that each of the following are true in the return value that you saved in the *act* section using the include assertion provided by Chai:

- To make sure that the method doesn't *remove* the wrong things
    - Assert that it contains the string " "
    - Assert that it contains the string " "
    - Assert that it contains the string "<select>"
    - Assert that it contains the string "</select>"

- To make sure that the method doesn't *add* the wrong things
    - Assert that it does not contain the string "<option>"
    - Assert that it does not contain the string "</option>"

- To make sure it replaces what you expect it to replace
    - Assert that it does not contain the string "<!-- Content here -->"

Run the test to make sure it passes.

Except for some string differences, this test will look nearly identical to the first test that you did for the `<li>` tags earlier in this project.

The second test

The second test reads

```
it("should return a single <option> for one category", () => {
  expect.fail('please write this test');
});
```

Replace the `expect.fail` line with a test that properly follows the *Three As* of unit testing.

In the *arrange* section, you will need to create an array for the `categories` argument that contains a single string and store it in a variable. You will use the variable in the action and the value that you typed in the assertion.

In the *act* section, you will invoke the `mergeCategories` function with the `template` as the first argument, the variable that contains the array with the single value as the second argument, and the string 'option' for the tag name as the third argument. Store the return value in a variable.

In the *assert* section, assert that each of the following are true using the include assertion provided by Chai:

- To make sure that the method doesn't *remove* the wrong things
    - Assert that it contains the string " "
    - Assert that it contains the string " "
    - Assert that it contains the string "<select>"
    - Assert that it contains the string "</select>"

- To make sure that the method *adds* the right things
    - Assert that it does contain the string "<option>your string here</option>" where "your string here" is the single value that you placed in the array

- To make sure it replaces what you expect it to replace
    - Assert that it does not contain the string "<!-- Content here -->"

Run the test to make sure it passes.

Except for some string differences, this test will look nearly identical to the second test that you did for the `<li>` tags earlier in this project.

The third test

The third test reads

```
it("should return an <option> for each category", () => {
  expect.fail('please write this test');
});
```

Replace the `expect.fail` line with a test that properly follows the *Three As* of unit testing.

In the *arrange* section, you will need to create an array for the `categories` argument that contains multiple strings and store it in a variable. You will use the variable in the action and the values that you typed in the assertion.

In the *act* section, you will invoke the `mergeCategories` function with the

`template` as the first argument, the variable that contains the array with the many values as the second argument, and the string 'option' for the tag name as the third argument. Store the return value in a variable.

In the *assert* section, assert that each of the following are true using the include assertion provided by Chai:

- To make sure that the method doesn't *remove* the wrong things
    - Assert that it contains the string " "
    - Assert that it contains the string " "
    - Assert that it contains the string "<select>"
    - Assert that it contains the string "</select>"

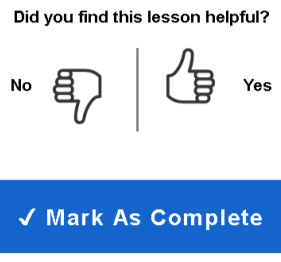
- To make sure that the method *adds* the right things, for *each* of the values that you put in your categories array:
    - Assert that it does contain the string "<option>value n</option>" where "value n" is one of the values in your array

- To make sure it replaces what you expect it to replace
    - Assert that it does not contain the string "<!-- Content here -->"

Run the test to make sure it passes.

Except for some string differences, this test will look nearly identical to the third test that you did for the `<li>` tags earlier in this project.

You have won this round!



Finished with this task? Click **Mark as Complete** to continue to the next page!