

CPSC1520 – JavaScript 4 Exercise: Making Decisions and Forms

Introduction

Validation of forms is everywhere when you create a JavaScript application and a web application in general. You want to ensure that you're getting correct and valid information from your users so that you can (for us in the future) save this information.

Exercise Step 1 – Adding a new Item to our Order

1. Link your JavaScript file in the HTML, do not change any HTML otherwise.
2. Select the form with the class "new-order-form"
3. Add an event listener on the form that will handle the "submit" event.
4. Using the event object, prevent the default action from happening
5. Assign the form elements from using the "elements" property on the "event.target" to variables with appropriate names.
6. Pass the form input **values** to the "addOrderItem" function given.
 - a. If you're ever confused about a specific variable, you can always use the console to print out the variables. The output should look something like this once you click the "Add to Order" button:

(*) Introduction to JavaScript

New Order

Item Name Price Size

Enter Item Name... Enter Price... Choose Size... ▼

Add to Order

coffee \$8

Medium

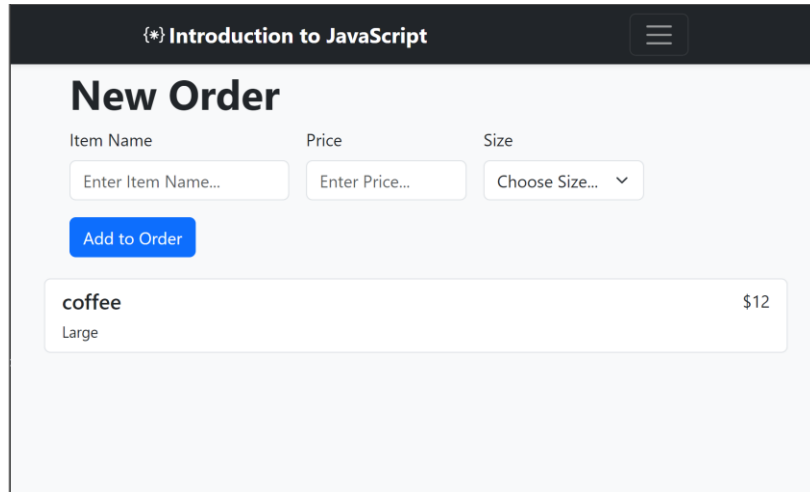
Exercise Step 2 – Validating Elements of the Form

1. There's a bug in our application. We get a blank row to our order if we submit our form with nothing in it. We only want populated rows because that is what we expect in our application.

The screenshot shows a web application titled "Introduction to JavaScript" with a "New Order" form. The form contains three input fields: "Item Name" (placeholder "Enter Item Name..."), "Price" (placeholder "Enter Price..."), and "Size" (a dropdown menu with "Choose Size..." and a downward arrow). Below these fields is a blue "Add to Order" button. Under the button is a table representing the order items. The table has two rows: the first row contains "coffee" and "\$8", and the second row is empty with a "\$" symbol in the price column.

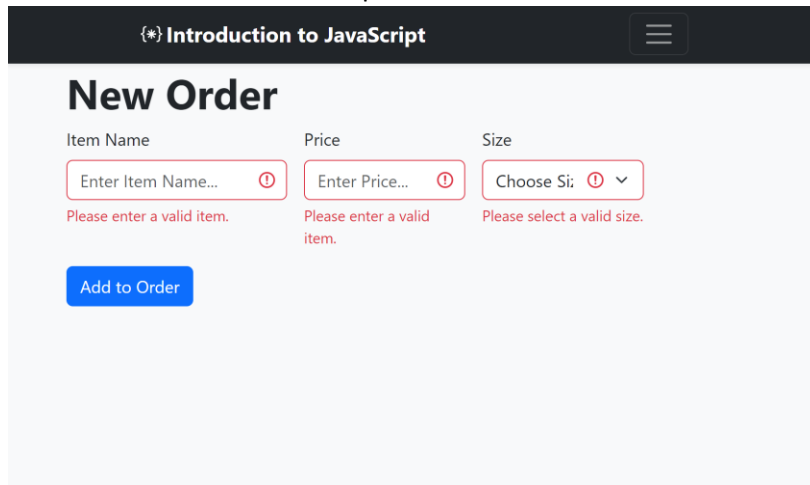
2. Create a variable named "isFormValid" that will be assigned a Boolean. Using this variable you, either execute or do not execute the "addOrderItem" with the values from the form.
3. In the following steps, use the "isFormValid" with the validation functions "isValueNotEmpty" and "isGreaterThanFive".
4. Validate the order item name.
 - a. If the order item name is not an empty string, then the item name is valid.
 - i. Remove the "is-invalid" class on the order item name using the elements' "classList.remove" function.
 - b. If the order item name is an empty string, then the item name is invalid.
 - i. Remove the "is-invalid" class on the order item name using the elements' "classList.add" function.
 - ii. Set the "isFormValid" to false.
5. Validate the order item price.
 - a. If the order item price is not an empty string, and if it's greater than five then the item price is valid.
 - i. Remove the "is-invalid" class on the order item price using the elements' "classList.remove" function.
 - b. If the order item price is an empty string, or if the price is less than five or if then the item price is invalid.
 - i. Remove the "is-invalid" class on the order item name using the elements' "classList.add" function.
 - ii. Set the "isFormValid" to false.
6. Validate the order size.
 - a. If the order size is not an empty string, then it's in valid.
 - i. Remove the "is-invalid" class on the order item size using the elements' "classList.remove" function.
 - b. If the order size is an empty string, then it's in valid.
 - i. Remove the "is-invalid" class on the order item size using the elements' "classList.add" function.

- ii. Set the “isFormValid” to false
7. Execute the “addOrderItem” function only if “isFormValid” is true. There are a couple of ways to do this, and either works.
8. You reset the values of each input on a **successful** form submission.
9. The functionality should look like the following result.
 - a. Successful result



The screenshot shows a web application titled "Introduction to JavaScript" with a hamburger menu icon. The main heading is "New Order". Below it, there are three input fields: "Item Name" with the placeholder "Enter Item Name...", "Price" with the placeholder "Enter Price...", and "Size" with a dropdown menu showing "Choose Size...". A blue "Add to Order" button is positioned below the "Item Name" field. Below the button, a summary box displays "coffee" as the item name, "Large" as the size, and "\$12" as the price.

- b. Invalid result with all of the inputs invalid.



The screenshot shows the same "New Order" form, but all three input fields are now invalid. Each field has a red border and a red error icon (a circle with an exclamation mark). Below each field is a red error message: "Please enter a valid item." for Item Name, "Please enter a valid item." for Price, and "Please select a valid size." for Size. The "Add to Order" button remains blue.

Test Cases

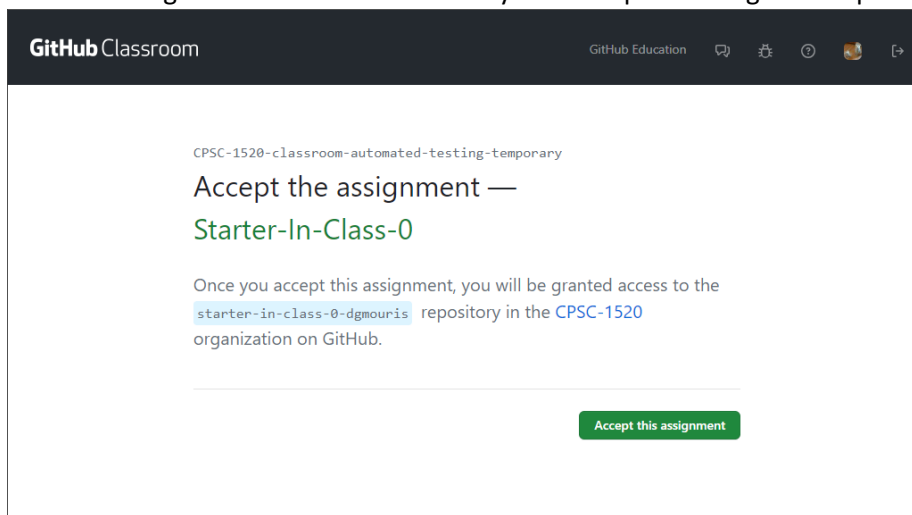
You might need clarification and need to know what values to test here to help you out.

- Test case: form invalid
 - Item Name: Burger
 - Price:
 - Size:
- test case: form invalid
 - Item Name:
 - Price: 18
 - Size:
- test case: form invalid

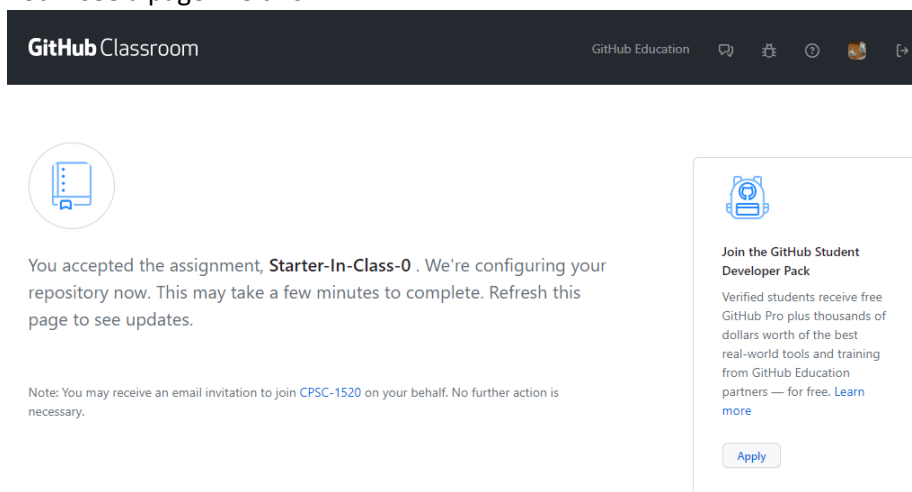
- Item Name: pop
- Price: 4
- Size: small
- test case: form invalid
 - Item Name:
 - Price:
 - Size: Small
- Test case: valid form
 - Item Name: pop
 - Price: 6
 - Size: small

Exercise Step 4 – Push up your code to github (accepting this assignment)

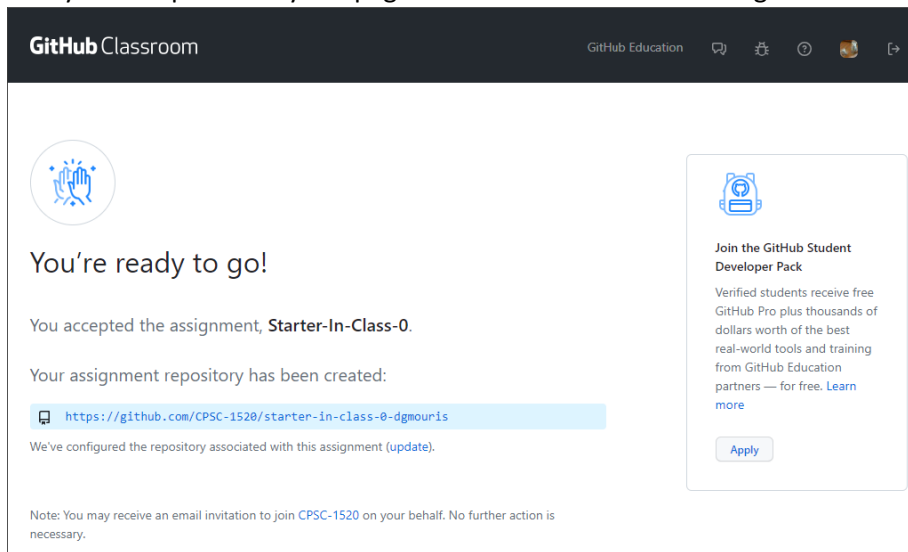
1. Open the link given and accept the assignment. Your link should look something like this. Note the image will be different because you'll accept the assignment specified.



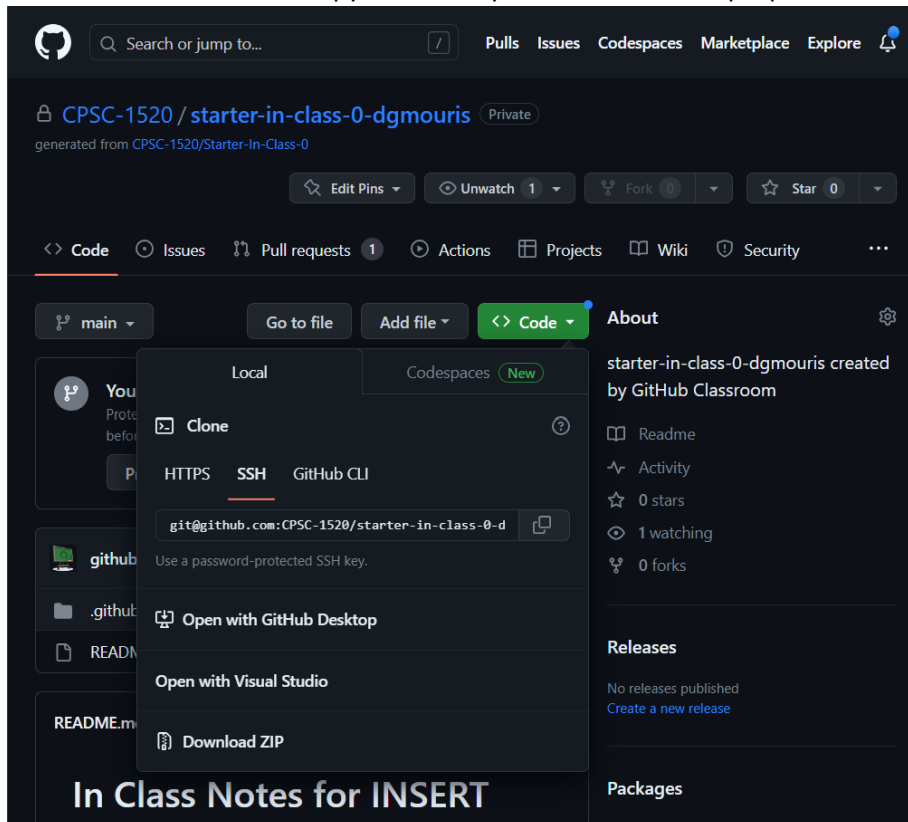
You'll see a page like this.



Once your repo is ready the page should look like the following.



2. You should see the page below once you click on the link highlighted in blue. Click the button that says "Code." You'll need to select "HTTPS" unless you've set up "SSH" (you can also set up GitHub CLI". Click on the copy icon once you've selected the proper icon.



3. Clone the repository in your console (or if you're using GitHub Desktop) using the "git clone REPO_URL" command.

```
DMOURIS@W309-DMORR2 C:\Users\dmouris\temp
$ git clone git@github.com:CPSC-1520/starter-in-class-0-dgmouris.git
Cloning into 'starter-in-class-0-dgmouris'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.

DMOURIS@W309-DMORR2 C:\Users\dmouris\temp
$
```

And go into this folder.

4. Make your changes, then add them to staging (using "git add .") and commit them (using "git commit -m "CHANGE THIS MESSAGE"). Once committed, push them up to GitHub (using "git push") it should look like below.

```
DMOURIS@W309-DMORR2 C:\Users\dmouris\temp\starter-in-class-0-dgmouris
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

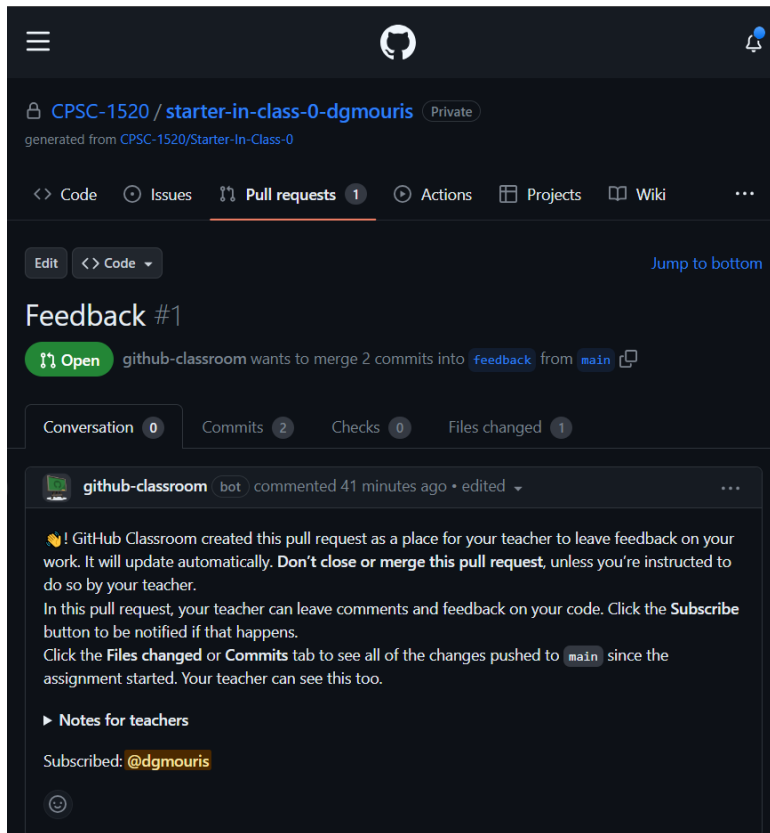
DMOURIS@W309-DMORR2 C:\Users\dmouris\temp\starter-in-class-0-dgmouris
$ git add README.md

DMOURIS@W309-DMORR2 C:\Users\dmouris\temp\starter-in-class-0-dgmouris
$ git commit -m "Made changes"
[main 9532c1b] Made changes
 1 file changed, 1 insertion(+), 3 deletions(-)

DMOURIS@W309-DMORR2 C:\Users\dmouris\temp\starter-in-class-0-dgmouris
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 373 bytes | 373.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:CPSC-1520/starter-in-class-0-dgmouris.git
   b6ef88e..9532c1b  main -> main

DMOURIS@W309-DMORR2 C:\Users\dmouris\temp\starter-in-class-0-dgmouris
$
```

5. If you click "Pull Requests" and then the first item called "Feedback" you should see your commit (seen at the bottom).



6. Upload the link of your repository to Moodle.

Grading

I'll give full marks if:

- All of the test cases pass.
- The form validates each input element separately.
- An invalid form submission should look like this:

- A valid form submission should look this:

Introduction to JavaScript

New Order

Item Name	Price	Size	
<input type="text" value="Enter Item Name..."/>	<input type="text" value="Enter Price..."/>	<input data-cs="2" data-kind="parent" type="text" value="Choose Size..."/>	
<input type="button" value="Add to Order"/>			
coffee		\$12	
Large			

If you don't follow the instructions specified you'll get a zero. There are no marks in between.