

## CPSC1520 – JavaScript 7 Exercise: ES Modules and Using open-source packages for validation.

### Introduction

A large part of JavaScript is using the ecosystem of open-source packages that are out there. In this assessment we're going to set up a project with parcel, use styles from bootstrap by importing it into our main.js, and use the validator.js package to validate our inputs.

### Exercise Step 1: Install packages using NPM and setup parcel.js

1. Open your command line and go to the folder of your extracted in-class-assessment.
2. In your terminal (in the same folder as your index.html) run the command "npm init" and just press enter until a **"package.json" file is in the same folder as your index.html file.**
3. Install the required packages that we'll need for this project. In the terminal run the following commands.
  - a. npm install validator
  - b. npm install bootstrap
  - c. npm install parcel -- save-dev

Note: You need to see this in your dependencies for your package.json file like so (except newer versions most likely):

```
"dependencies": {  
  "bootstrap": "^5.3.2",  
  "parcel": "^2.10.2",  
  "validator": "^13.11.0"  
}
```

4. In your package.json replace the scripts from what's there to the following.

From:

```
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1"  
},
```

To this (remove "main" attribute in your package.json):

```
"source": "index.html",  
"scripts": {  
  "start": "parcel",  
  "build": "parcel build"  
},
```

5. When running your parcel web build with **"npm run start"** and opening the site at the port that parcel gives you you should see the following site. NOTE: **If you're using liveserver from vs code**

for the rest of this assessment it just won't work.

About

Our app to learn javascript.

Contact

{\*} Introduction to JavaScript

## Sign up!

Email address

Please enter a valid username

Password

Please enter a strong password:

- Minimum Length: 8
- Minimum Lowercase: 1
- Minimum Uppercase: 1
- Minimum Numbers: 1
- Minimum Symbols: 1

Please retype Password

sorry the passwords do not match

Sign in

## Welcome

6. In your "js" folder create an "main.js" file and link it in your html as follows.

```
<script type="module" src="js/main.js"></script>
```

Note: To ensure that your main.js is built and is using it successfully you should just display something for yourself. Only after you've done these steps you should move to the next step.

Exercise Step 2: Import your CSS into your main.js file to apply styles.

1. In your main.js import bootstrap so that you have some default styles. Referring to bootstraps' documentation you should be able to import it ([reference here](#)) as the first thing in your main.js like so:

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

Note: If everything is working successfully it should look like the image below (If it doesn't refer back to the previous steps):

{\*} Introduction to JavaScript

## Sign up!

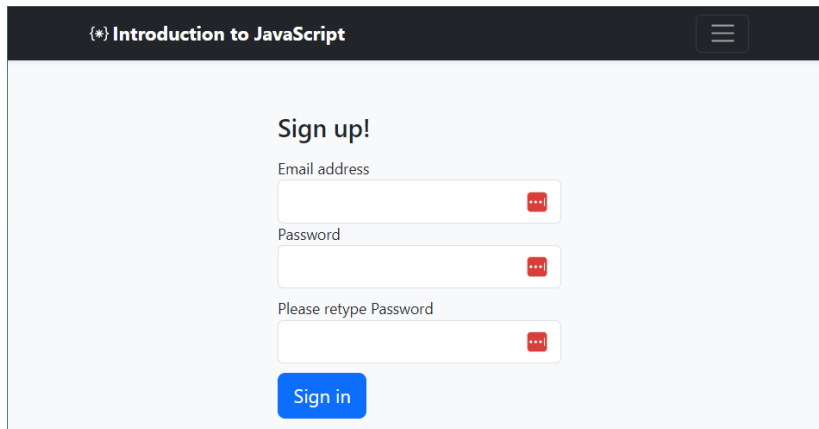
Email address

Password

Please retype Password

Sign in

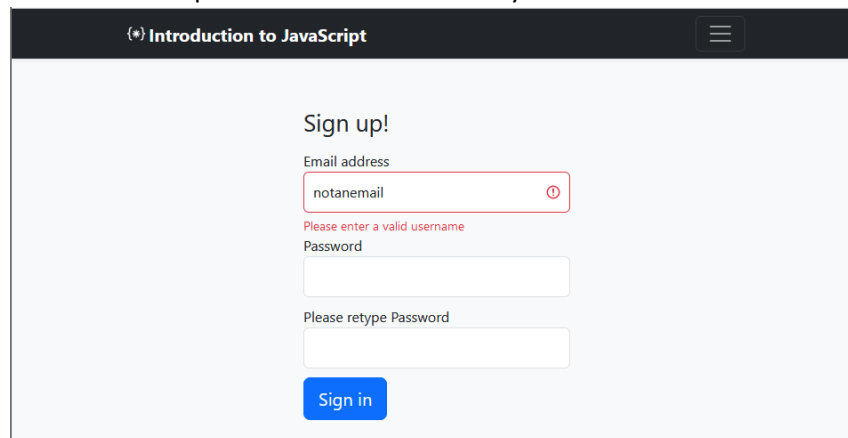
2. Using **relative file paths** import your **signin.css** file to your **main.js**. Once you do so your sign up form should be a little narrower and look like the screen shot below.



The screenshot shows a web application header with the text "Introduction to JavaScript" and a hamburger menu icon. Below the header is a "Sign up!" form. The form contains three input fields: "Email address", "Password", and "Please retype Password". Each input field has a red "x" icon on the right side, indicating a validation error. At the bottom of the form is a blue "Sign in" button.

### Exercise Step 3: Use Validator.js in the event listener to display the errors.

1. Add an event listener on the "form".
  - a. Get all of the input elements, this would be the input for the: email address/username, password one and password two
3. Import the validator package that you installed in step 1.
  - a. You can refer [here to the documentation to see how to import validator using es6](#)
4. Using the server side validation techniques we've used in the past ([refer to docs here](#)) and validator validate all of the inputs in the form.
  - a. **Validate the email using the "isEmail" function** (which returns a boolean) under validators ([docs here](#)) and add/remove the "is-invalid" class when appropriate. Here's an example of what it should ideally look like without an email.



The screenshot shows the same "Sign up!" form as before, but with a validation error. The "Email address" field now contains the text "notanemail" and has a red border and a red "x" icon. Below the field, the text "Please enter a valid username" is displayed in red. The "Password" and "Please retype Password" fields are empty, and the "Sign in" button is still present at the bottom.

- b. **Validate the password one with the "isStrongPassword" function** (with no options using default, [refer to docs here](#)) and add/remove the "is-invalid" class when appropriate.

Here's what it should look like with a valid email address and an invalid password:

The screenshot shows a web application header with the text "{\*} Introduction to JavaScript" and a hamburger menu icon. The main content area is titled "Sign up!". It contains two input fields: "Email address" with the value "dmouris@nait.ca" and "Password" which is empty and has a red border with an information icon. Below the password field, there is a red error message: "Please enter a strong password:" followed by a bulleted list of requirements: "Minimum Length: 8", "Minimum Lowercase: 1", "Minimum Uppercase: 1", "Minimum Numbers: 1", and "Minimum Symbols: 1". Below this list is a "Please retype Password" label and an empty input field. At the bottom is a blue "Sign in" button.

- c. **Validate the password two by checking if it's equal to the first password** and add/remove the "is-invalid" class when appropriate.

Here's what it should look like when the email is valid, the password one is valid but password two is not equal to password one:

The screenshot shows the same web application header. The "Sign up!" section has the "Email address" field filled with "dmouris@nait.ca". The "Password" field is filled with ten dots and has a red border with an information icon. The "Please retype Password" label is above an input field that contains one dot and also has a red border with an information icon. Below this field is a red error message: "sorry the passwords do not match". At the bottom is a blue "Sign in" button.

5. If all of the form inputs and the user clicks submit, call the "signInSuccess" function with the email provided. The application should look like the screenshot below.

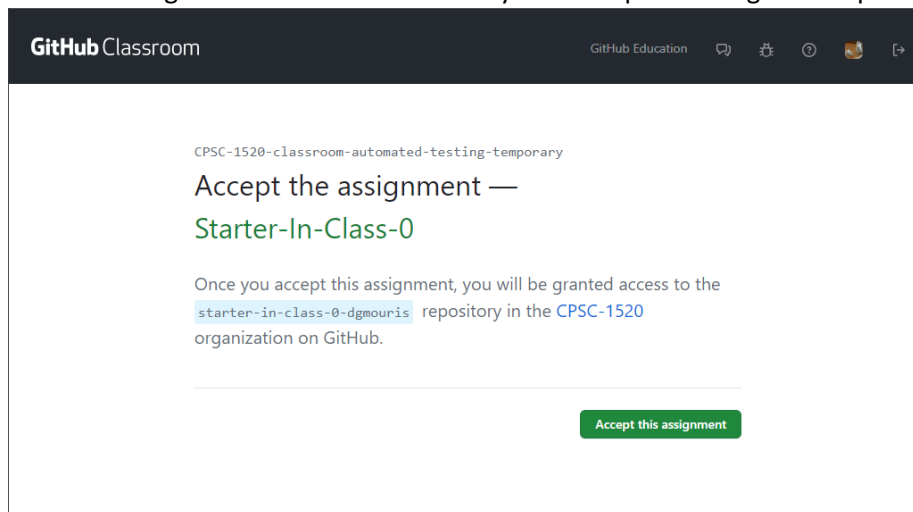
The screenshot shows the web application header with "{\*} Introduction to JavaScript" and a hamburger menu icon. The main content area displays a large "Welcome dmouris@nait.ca!" message.

Test Cases: All of these should pass.

- Valid Case.
  - o email address: dmouris@nait.ca
  - o password one: GarySteve1@
  - o password two: GarySteve1@
- Invalid Case: passwords don't match, but are strong
  - o email address: dmouris@nait.ca
  - o password one: GarySteve1@
  - o password two: Something1@
- Invalid Case: passwords don't match, and are not strong
  - o email address: dmouris@nait.ca
  - o password one: something
  - o password two: terriblepassword
- Invalid Case: passwords match, but are not strong
  - o email address: dmouris@nait.ca
  - o password one: something
  - o password two: something
- Invalid Case: email isn't valid.
  - o email address: supercoolperson
  - o password one: GarySteve1@
  - o password two: GarySteve1@

#### Exercise Step 4 – Push up your code to GitHub


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.


**GitHub Classroom**

GitHub Education



You accepted the assignment, **Starter-In-Class-0**. We're configuring your repository now. This may take a few minutes to complete. Refresh this page to see updates.

Note: You may receive an email invitation to join [CPSC-1520](#) on your behalf. No further action is necessary.

**Join the GitHub Student Developer Pack**


Verified students receive free GitHub Pro plus thousands of dollars worth of the best real-world tools and training from GitHub Education partners — for free. [Learn more](#)

[Apply](#)

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

**GitHub Classroom**


GitHub Education



## You're ready to go!


You accepted the assignment, **Starter-In-Class-0**.

Your assignment repository has been created:

 <https://github.com/CPSC-1520/starter-in-class-0-dgmouris>

We've configured the repository associated with this assignment ([update](#)).

Note: You may receive an email invitation to join [CPSC-1520](#) on your behalf. No further action is necessary.

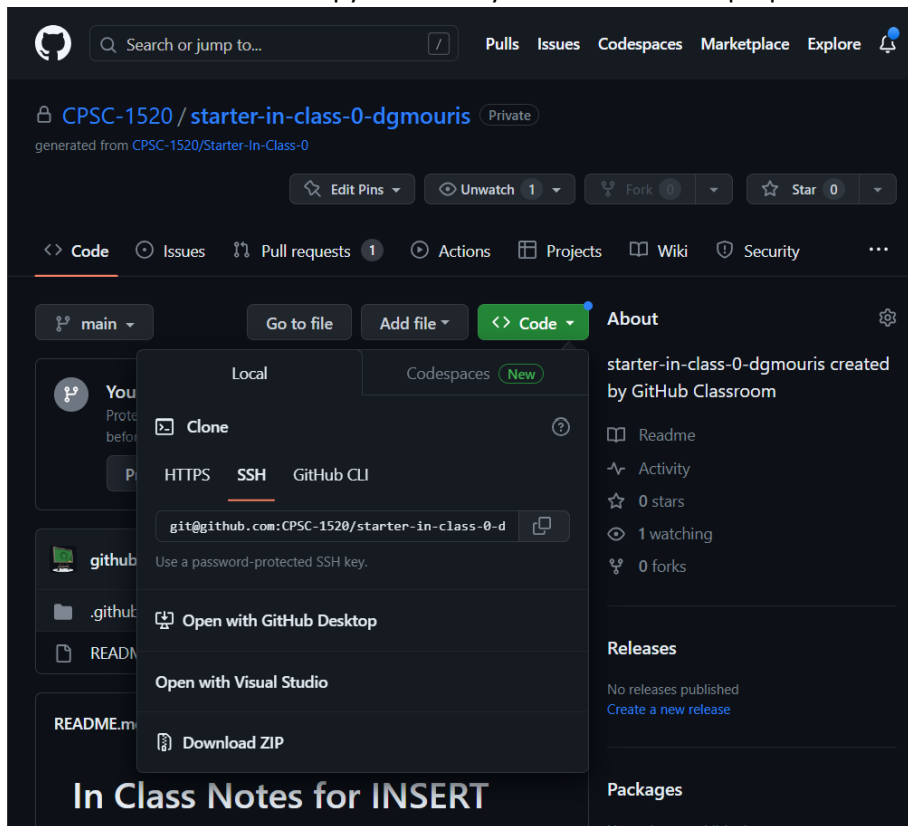
**Join the GitHub Student Developer Pack**

Verified students receive free GitHub Pro plus thousands of dollars worth of the best real-world tools and training from GitHub Education partners — for free. [Learn more](#)

[Apply](#)

- 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

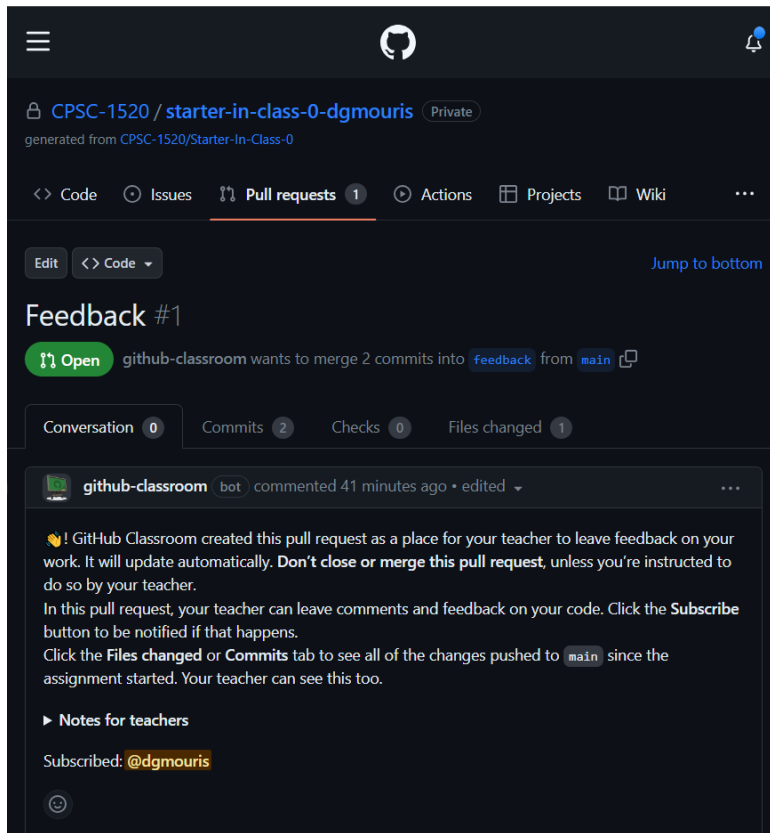
DMOURIS@W309-DMORR2 C:\Users\dmouris\temp\starter-in-class-0-dgmouris
$ git commit -m "Made changes"
[main 9532c1b] Made changes
 1 file changed, 1 insertion(+), 3 deletions(-)

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

- The test cases should be valid or invalid based on the data provided and should match the screenshots provided.
- If valid the welcome screenshot should be shown.
- Grading will be either pass/fail or complete/partially complete/incomplete based on your instructor preference.