Lab 3

Due Feb 21, 2023 by 11:59pm **Points** 100 **Submitting** a file upload

CS-546 Lab 3

The purpose of this lab is to familiarize yourself with asynchronous programming in JavaScript, as well as using modules from the Node.js Package Manager (npm (https://www.npmjs.com/)).

For this lab, you **must** use the `async/await` keywords (not Promises). You will also be using `axios` (https://github.com/axios/axios) , which is a HTTP client for Node.js; you can install it with `npm i axios`.

In addition, you must have error checking for the arguments of all your functions. If an argument fails error checking, you should throw a string describing which argument was wrong, and what went wrong.

You will be creating three `.js` files: `movies.js`, `users.js` and `app.js`.

You can download the starter template here: lab3_stub.zip (https://sit.instructure.com/courses/64637/files/11169560?wrap=1) (https://sit.instructure.com/courses/64637/files/11169560/download?download_frd=1) **PLEASE NOTE: THE STUB DOES NOT INCLUDE THE PACKAGE.JSON FILE. YOU WILL NEED TO CREATE IT! DO NOT FORGET TO ADD THE START COMMAND AND "type": "module". DO NOT ADD ANY OTHER FILE OR FOLDER APART FROM PACKAGE.JSON FILE.**

Note: Remember that the order of the keys in the objects does not matter so `{firstName: "Patrick", lastName: "Hill"}` is the same as: `{lastName: "Hill", firstName: "Patrick"}`

Network JSON Data

You will be downloading JSON files from the following GitHub Gists:

- movies.json (Links to an external site.) (https://gist.githubusercontent.com/jdelrosa/78dfa36561d5c06f7e62d8cce868cf8e/raw/2292be808f74c9486d408
- users.json (Links to an external site.) (https://gist.githubusercontent.com/jdelrosa/381cbe8fae75b769a1ce6e71bdb249b5/raw/564a41f84ab00655524a

For every function you write, you will download the necessary JSONs with `axios`. DO NOT just save the data into a local file, you MUST use Axios to get the data. Here is an example of how to do so:

```
async function getMovies(){
  const { data } = await axios.get('https://URL_FROM_GIST_HERE/movies.json')
  return data // this will be the array of user objects
}
```

Instead of making the request in every single function, remember that code reuse is key, so if you see you are making the same axios request in all of your functions, it's best to put it in a function like noted above and then calling that function in all the functions that need to get the data from whichever json file you're working with. Always do this when you see you are doing the same thing over and over again in multiple different places. It's much easier to maintain. Say if the URL of the file ever changes, then you only need to change it in one place, not 10 different places.

users.js

This file will export the following functions:

getUserById(id)

This will return the user object for the specified id within the `users.json` array. Note: The `id` is case sensitive.

You must check:

- That the `id` parameter exists and is of proper type (string). If not, throw an error.
- If the id exists and is in the proper type but the `id` is not found in the array of users, throw a 'user not found' error.
- if the `id` parameter is just empty spaces, throw an error.

```
await getUserById("48fded55-37cd-4e6b-8f19-e78b481a14a4");
\\ Returns:
{id:"48fded55-37cd-4e6b-8f19-e78b481a14a4",username:"abrett0",password:"YQ8Jpot33Mf",first_name:"Abigail",last_name:"Brett",email:"abrett0@gizmodo.com",favorite_genre:"Fantasy"}

await getUserById(-1); \\ Throws Error
await getUserById(1001); \\ Throws Error
await getUserById();\\ Throws Error
await getUserById('7989fa5e-5617-43f7-a931-46036f9dbcff');\\ Throws user not found Error
```

sameGenre(genre)

For this function, you will return an array of the first 50 users (so the 50 that appear first in the array of users. If there are less than 50, then just return however many there are) who have the same favorite genre from `users.json`. The array will be comprised of the names of user ("firstName lastName" format) sorted alphabetically by **last name**. You must return at least two users, so if there are not 2 or more users that have the same favorite genre for the `genre` provided you will throw an error.

You must check:

- That the `genre` parameter exists and is of proper type (string). If not, throw an error.

- If there are not at least two users that have the same favorite genre as the `genre` provided , you will throw an error.
- if `genre` is just empty spaces, throw an error.
- The `genre` parameter must be case in-sensitive i.e. `sameGenre("action")` should return the same results as passing `sameGenre("ACTION")`

```
await sameGenre("Action");
\\ Returns:
['Shay Claydon', 'Merridie Confort', 'Bent Crowest', 'Shurlocke Cull', 'Lonny Dechelle', 'Olia Shefton']

await sameGenre(); \\ Throws Error
await sameGenre("IMAX"); \\ Throws Error since there are not two people with that genre
await sameGenre(123); \\ Throws Error
await sameGenre(["Action"]); \\ Throws Error
await sameGenre(true); \\ Throws Error
```

moviesReviewed(id)

This function will take in `id` of a user object and return an array of all the movies that the specified user left a review on. The array will be comprised of `objects` of the following format: `{title: {review object with username, rating, and review properties as shown below}}`.

To get the movie a user has reviewed, use the `username` field found in both the user object in `users.json` and movie object in `movies.json.` So this takes in an ID of the user in users.json, gets the username for that user and then finds all the movies that user reviewed.

Note: The id is case-sensitive.

- That the `id` parameter exists and is of proper type (string). If not, throw an error.
- If the `id` exists and is in the proper type but the `id` is not found in the array of users, throw a 'user not found' error.
- if the `id` parameter is just empty spaces, throw an error.

```
await moviesReviewed('64035fad-a5b7-48c9-9317-3e31e22fe26c')
\\ Returns:
[{'Charlie's Angels': {username:"cfinkle5",rating:4,review:"Solid, good movie."} },
{'Class of 1999 II: The Substitute': {username:"cfinkle5",rating:4,review:"Solid, good movie."} },
{'Terminator 3: Rise of the Machines': {username:"cfinkle5",rating:2,review:"It was meh, plot was very bad."} }]

await moviesReviewed(-1); \\ Throws Error
await moviesReviewed(1001); \\ Throws Error
await moviesReviewed();\\ Throws Error
await moviesReviewed('7989fa5e-5617-43f7-a931-46036f9dbcff');\\ Throws user not found Error
```

referMovies(id)

This function will take the `id` of a user object and return an array of all the movies that match that user's favorite genre that they have NOT reviewed previously. The idea is this function will recommend movies to the user that they have not reviewed(we assume they haven't seen the movies they haven't reviewed) that are listed in their favorite genre. The array will contain the titles of the movies. **NOTE:** Please look at the genre data in movies.json, you'll notice that some movies have more than one genre separated by a pipe character.

You will need to return the movie, as long as the user's favorite genre is included in the genre field. For example, some movies have their genre like this: "genre": "Drama|Musical", If the user's favorite genre is "Drama" all the movies that have the genre of "Drama" should be returned as well as any movies where "Drama" is contained in the genre field like the "Drama|Musical" example above.

Note: The id is case-sensitive.

You must check:

- That the `id` parameter exists and is of proper type (string). If not, throw an error.
- If the `id` exists and is in the proper type but the `id` is not found in the array of users, throw a 'user not found' error.
- if the `id` parameter is just empty spaces, throw an error

```
await referMovies('5060fc9e-10c7-4f38-9f3d-47b7f477568b'); \\ Returns:
['Fly Me to the Moon', 'Gravity', 'Spiderwick Chronicles, The', 'How to Train Your Dragon', 'Wings of Courage', 'Ha
ppy Feet Two']

await referMovies(-1); \\ Throws Error
await referMovies('       '); \\ Throws Error
await referMovies(); \\ Throws Error
await referMovies('7989fa5e-5617-43f7-a931-46036f9dbcff'); \\ Throws user not found Error
```

movies.js

This file will export the following three functions:

findMoviesByDirector(directorName)

For this function, you will return an array of `objects` that contains all the movies that were directed by `directorName` provided.

You must check:

- That `directorName` parameter exists and is of the proper type (string). If not, throw an error.
- You must check to make sure the `directorName` parameter is not just empty spaces: If it is, throw an error.
- If the no movies can be found in `movies.json` for the `directorName` provided, then throw an error.

```
findMoviesByDirector("Fernando Dollimore") Would return:
[
  {
    id: '040d7398-136c-45f0-89b8-9b73c67c617e',
    title: 'Company',
    genre: 'Drama|Musical',
    director: 'Fernando Dollimore',
    release_date: '10/27/2020',
    runtime: '1h 14mins',
    mpa_rating: 'PG-13',
    cast: ['Huberto Snoddon', 'Horacio Scoggins'],
    streaming_service: {
      company: 'Netflix',
      link: 'https://Netflix.com/Company'
    },
    reviews: [
      {
        username: 'jsorrelaw',
        rating: 2,
        review: 'It was meh, plot was very bad.'
      },
      {username: 'sgiacobo1n', rating: 3, review: 'A very ok movie.'},
      {username: 'egrigolieb', rating: 3, review: 'A very ok movie.'},
      {username: 'lmcinnesmk', rating: 4, review: 'Solid, good movie.'}
    ]
  },
  {
    id: 'e8b006a5-8a81-4718-ae52-11b2bd02f741',
    title: 'Flashbacks of a Fool',
    genre: 'Drama',
    director: 'Fernando Dollimore',
    release_date: '07/15/2010',
    runtime: '2h 58mins',
    mpa_rating: 'PG',
    cast: [
      'Iver Hubbucks',
      'Tandi Arminger',
      'Willette Furze',
      'Feliks Edowes',
      'Neddie Ashleigh'
    ],
    streaming_service: {
      company: 'Paramount+',
      link: 'https://Paramount+.com/Flashbacks of a Fool'
    },
    reviews: [
      {
        username: 'tjoice3z',
        rating: 2,
        review: 'It was meh, plot was very bad.'
      },
      {
        username: 'lhumpherstonjo',
        rating: 2,
        review: 'It was meh, plot was very bad.'
      },
      {username: 'sgiacobo1n', rating: 1, review: 'HORRIBLE MOVIE!!!'},
      {username: 'kcoumbe9m', rating: 3, review: 'A very ok movie.'}
    ]
  },
  {
    id: 'f77972aa-9fdf-4465-9948-ba4acfea4d16',
    title: 'Last Time, The',
    genre: 'Comedy|Drama|Romance',
    director: 'Fernando Dollimore',
    release_date: '05/24/2013',
    runtime: '3h 32mins',
    mpa_rating: 'R',
    cast: ['Isaiah Gabbett', 'Merrili Maud', 'Raynard Tuxsell'],
    streaming_service: {
      company: 'Peacock',
      link: 'https://Peacock.com/Last Time, The'
    },
    reviews: [
      {
        username: 'lbickelll',
        rating: 2,
        review: 'It was meh, plot was very bad.'
      },
      {
        username: 'abuttersm2',
        rating: 5,
        review: 'OMG I loved it. AMAZING 10/10!!!!'
      }
    ]
  },
  {
    id: 'bcafe739-d928-4440-b3a9-4cc554a1cb2a',
    title: 'Rambo III',
    genre: 'Action|Adventure|Thriller|War',
    director: 'Fernando Dollimore',
    release_date: '02/11/2020',
    runtime: '1h 16mins',
    mpa_rating: 'R',
    cast: ['Meier Craine', 'Lorrie Yanin', 'Nertie Kadar', 'Pattie Caffin'],
    streaming_service: {
      company: 'HBO Max',
      link: 'https://HBO Max.com/Rambo III'
    },
    reviews: [
      {
        username: 'jjackettcr',
        rating: 5,
        review: 'OMG I loved it. AMAZING 10/10!!!!'
      },
      {
        username: 'bboziermu',
        rating: 2,
        review: 'It was meh, plot was very bad.'
      },
      {
        username: 'apergensrj',
        rating: 2,
        review: 'It was meh, plot was very bad.'
      },
      {username: 'cempsbj', rating: 4, review: 'Solid, good movie.'}
    ]
  }
];
```

findMoviesByCastMember(castMemberName)

For this function, you will return an array of `objects` that contains all the movies where the `castMemberName` provided has starred in.

You must check:

- That `castMemberName` parameter exists and is of the proper type (string). If not, throw an error.
- You must check to make sure the `castMemberName` parameter is not just empty spaces: If it is, throw an error.
- If the no movies can be found in `movies.json` for the `castMemberName` provided, then throw an error.

```
findMoviesByCastMember("Huberto Snoddon") Would return:
[
  {
    id: '040d7398-136c-45f0-89b8-9b73c67c617e',
    title: 'Company',
    genre: 'Drama|Musical',
    director: 'Fernando Dollimore',
    release_date: '10/27/2020',
    runtime: '1h 14mins',
    mpa_rating: 'PG-13',
    cast: ['Huberto Snoddon', 'Horacio Scoggins'],
    streaming_service: {
      company: 'Netflix',
      link: 'https://Netflix.com/Company'
    },
    reviews: [
      {
        username: 'jsorrelaw',
        rating: 2,
        review: 'It was meh, plot was very bad.'
      },
      {username: 'sgiacobo1n', rating: 3, review: 'A very ok movie.'},
      {username: 'egrigolieb', rating: 3, review: 'A very ok movie.'},
      {username: 'lmcinnesmk', rating: 4, review: 'Solid, good movie.'}
    ]
  },
  {
    id: 'ab000bf0-f2e5-4cda-9294-e588a734f0ef',
    title: "Herod's Law (Ley de Herodes, La)",
    genre: 'Comedy|Crime|Mystery',
    director: 'Lise Glanister',
    release_date: '06/13/2003',
    runtime: '1h 57mins',
    mpa_rating: 'NC-17',
    cast: ['Huberto Snoddon', 'Mickie Rankine'],
    streaming_service: {
      company: 'Amazon Prime Video',
      link: "https://Amazon Prime Video.com/Herod's Law (Ley de Herodes, La)"
    },
    reviews: [{username: 'iaistonli', rating: 1, review: 'HORRIBLE MOVIE!!!'}]
  }
];
```

getOverallRating(title)

Given the `title` provided, you will calculate the `overallRating` of that specified movie object from `movies.json`. The `overallRating` will be calculated by taking the average of all the review ratings of the movie. The `overallRating` should go up to 1 decimal place. For rounding, you can use the `Math.floor()` function **https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/floor (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/floor)**

You must check:

- That `title` parameter exists and is of the proper type (string). If not, throw an error.
- You must check to make sure the `title` parameter is not just empty spaces: If it is, throw an error.
- If the movie cannot be found in `movies.json` for the supplied `title` parameter, then throw an error.

```
await getOverallRating('Asterix and the Vikings (Astérix et les Vikings)');
\\ Returns: 2.2


await getOverallRating(43); \\ Throws Error
await getOverallRating(' '); \\ Throws error
await getOverallRating('Mamma Mia'); \\ Throws error No movie with that title
await getOverallRating(); \\ Throws Error
```

getMovieById(id)

This will return the Movie for the specified id within the `movies.json` array. Note: The `id` is case sensitive.

You must check:

- That the `id` parameter exists and is of proper type (string). If not, throw an error.
- If the id exists and is in the proper type but the `id` is not found in the array of movies, throw a 'movie not found' error.
- if the `id` parameter is just empty spaces, throw an error.

```
await getMovieById("38fd6885-0271-4650-8afd-6d09f3a890a2");
\\ Returns:
{
   id:"38fd6885-0271-4650-8afd-6d09f3a890a2",
   title:"Asterix and the Vikings (Astérix et les Vikings)",
   genre:"Adventure|Animation|Children|Comedy|Fantasy",
   director:"Charissa Edinboro",
   release_date:"06/29/2007",
   runtime:"2h 35mins",
   mpa_rating:"R",
   cast:["Sharl Covert","Ailyn Howcroft","Nissie Henrys"],
   streaming_service:{company:"Disney+",link:"https://Disney+.com/Asterix and the Vikings (Astérix et les Viking
s)"},
   reviews:[{username:"afrill27",rating:3,review:"A very ok movie."},
              {username:"tchedzoy2v",rating:1,review:"HORRIBLE MOVIE!!!"},
              {username:"ltruckettim",rating:2,review:"It was meh, plot was very bad."},
              {username:"fgoodale6l",rating:3,review:"A very ok movie."}]
}

await getMovieById(-1); \\ Throws Error
await getMovieById(1001); \\ Throws Error
await getMovieById();\\ Throws Error
await getMovieById('7989fa5e-5617-43f7-a931-46036f9dbcff');\\ Throws movie not found Error
```

app.js

This file is where you will import your functions from the two other files and run test cases on your functions by calling them with various inputs. We will not use this file for grading and is only for your testing purposes to make sure:

1. Your functions in your 2 files are exporting correctly.

2. They are returning the correct output based on the input supplied (throwing errors when you're supposed to, returning the right results etc..).

Note: You will need an `async` function in your app.js file that `awaits` the calls to your function like the example below. You put all of your function calls within `main` each in its own `try/catch` block. and then you just call `main()`.

```
import * as movies from "./movies.js";

async function main(){
    try{
        const moviedata = await movies.getMovies();
        console.log (moviedata);
    }catch(e){
        console.log (e);
    }
}

//call main
main();
```

Requirements

1. Write each function in the specified file and export the function so that it may be used in other files.
2. Ensure to properly error check for different cases such as arguments existing and of the proper type as well as **throw (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/throw)** if

anything is out of bounds such as invalid array index or negative numbers for different operations.

3. Submit all files (including `package.json`) in a zip with your name in the following format: `LastName_FirstName.zip`.
4. Make sure to save any npm packages you use to your `package.json.`
5. **DO NOT** submit a zip containing your `node_modules` folder.

Lab 3 Rubric Spring 2023

Criteria	Ratings		Pts
users.js - getUserById Test cases used for grading will be different from assignment examples.	**8 to >0.0 pts** **1 Points/Error Handling Test Case & 2 Points/Valid Input Test Case** 4 Error Handling Test Cases & 2 Valid Input Test Cases. The function is implemented correctly, and few or all test cases pass.	**0 pts** **All Test Cases Failed** Incorrect implementation or none of the test cases pass.	8 pts
users.js - sameGenre Test cases used for grading will be different from assignment examples.	**14 to >0.0 pts** **1 Points/Error Handling Test Case & 2 Points/Valid Input Test Case** 6 Error Handling Test Cases & 4 Valid Input Test Cases. The function is implemented correctly, and few or all test cases pass.	**0 pts** **All Test Cases Failed** Incorrect implementation or none of the test cases pass.	14 pts
users.js - moviesReviewed Test cases used for grading will be different from assignment examples.	**14 to >0.0 pts** **2 Points/Error Handling Test Case & 5 Points/Valid Input Test Case** 6 Error Handling Test Cases & 4 Valid Input Test Cases. The function is implemented correctly, and few or all test cases pass.	**0 pts** **All Test Cases Failed** Incorrect implementation or none of the test cases pass.	14 pts
users.js - referMovies Test cases used for grading will be different from assignment examples.	**14 to >0.0 pts** **2 Points/Error Handling Test Case & 5 Points/Valid Input Test Case** 6 Error Handling Test Cases & 4 Valid Input Test Cases. The function is implemented correctly, and few or all test cases pass.	**0 pts** **All Test Cases Failed** Incorrect implementation or none of the test cases pass.	14 pts

Criteria	Ratings		Pts
movies.js - findMoviesByDirector Test cases used for grading will be different from assignment examples.	**14 to >0.0 pts** **2 Points/Error Handling Test Case & 4 Points/Valid Input Test Case** 6 Error Handling Test Cases & 4 Valid Input Test Cases. The function is implemented correctly, and few or all test cases pass.	**0 pts** **All Test Cases Failed** Incorrect implementation or none of the test cases pass.	14 pts
movies.js - findMoviesByCastMember Test cases used for grading will be different from assignment examples.	**14 to >0.0 pts** **1 Points/Error Handling Test Case & 2 Points/Valid Input Test Case** 6 Error Handling Test Cases & 4 Valid Input Test Cases. The function is implemented correctly, and few or all test cases pass.	**0 pts** **All Test Cases Failed** Incorrect implementation or none of the test cases pass.	14 pts
movies.js - getMovieById Test cases used for grading will be different from assignment examples.	**8 to >0.0 pts** **1 Points/Error Handling Test Case & 2 Points/Valid Input Test Case** 4 Error Handling Test Cases & 2 Valid Input Test Cases. The function is implemented correctly, and few or all test cases pass.	**0 pts** **All Test Cases Failed** Incorrect implementation or none of the test cases pass.	8 pts
movies.js - getOverallRating Test cases used for grading will be different from assignment examples.	**14 to >0.0 pts** **1 Points/Error Handling Test Case & 2 Points/Valid Input Test Case** 6 Error Handling Test Cases & 4 Valid Input Test Cases. The function is implemented correctly, and few or all test cases pass.	**0 pts** **All Test Cases Failed** Incorrect implementation or none of the test cases pass.	14 pts

Criteria	Ratings		Pts
			Total Points: 100