APIs

For Part 1 and Part 2 of this lab, keep track of your answers in a regular document (Word, Google Doc, Pages).

Part 1: SWAPI

Instructions

For each of the following, what is the complete URL(s) (including params or queries) that you need to go to in order to reach the following data:

1. the height of Darth Vader: https://swapi.dev/api/people/4, https://swapi.dev/api/people?search=darth

2. the population of the planet Alderaan: `http://swapi.dev/api/planets/2/`

3. the name of the manufacturer of the Millennium Falcon: `http://swapi.dev/api/starships/10/`

4. the name of the species that C-3PO belongs to (multiple URLs):
   `http://swapi.dev/api/species/2/`

   `http://swapi.dev/api/people/2/`

5. the title of each film that Obi-Wan Kenobi is in (multiple URLs):
   `http://swapi.dev/api/people/10/`
6. `"http://swapi.dev/api/films/1/",`
7. `"http://swapi.dev/api/films/2/",`
8. `"http://swapi.dev/api/films/3/",`
9. `"http://swapi.dev/api/films/4/",`
10. `"http://swapi.dev/api/films/5/",`
11. `"http://swapi.dev/api/films/6/"`
12.
13. use the search query (the how to on the search query is at the bottom of the Getting Started section of the documentation) to get the information about the Millennium Falcon, it's a starship:

    https://swapi.dev/api/starships?search=millennium

Part 2: Social Mountain

Summary

In this section, you'll be looking through the documentation for the Social Mountain API and answering questions. You'll also be making requests and recording the URLs and some information about the responses. Run the requests in Postman. **Note: this API is live and viewable by your classmates and staff. Keep things appropriate for class.**

You can view the documentation for the Social Mountain API here

The base URL of your requests is: https://practiceapi.devmountain.com/api (make sure to have the "s" in "https")

1. Check if the POST request accept params, queries, and/or a body. Which one(s) and what information is it expecting to be sent?
    a. Required body
2. What data type does the GET request return?
    a. Array of all posts
3. What would the URL look like for deleting the post with the id 555? (This post does not exist anymore, but the syntax is the same for existing posts,)
    a. https://practiceapi.devmountain.com/api/posts?id=555
4. List the possible response codes from the GET request at '/posts/filter'
    a. 200
    b. 409
5. Create a post whose text is your name, record the URL and body here:
    a. https://practiceapi.devmountain.com/api/posts?id=6620
    b. {
    c. "text": "Annie"
    d. }

6. What would the URL and body object be to update the post you just made to contain your faovrite color instead of your name?

    a. https://practiceapi.devmountain.com/api/posts?id=6620 (in put)
7. What is the URL to get posts that contain the text “blue”?
    a. https://practiceapi.devmountain.com/api/posts/filter?text=blue
8. Make a request to GET all the posts. What are the content type and charset of the response? (Hint: look on the Headers)
    a. application/json; charset=utf-8
9. What would cause a PUT request to return a 409 status?
    a.

Request was missing req.query.id or req.body.text

10. What happens if you try to send a query in the GET request URL? Why do you get that response?
    a. Request was missing req.query.id or req.body.text
    b. Get request has to include query id or body text

Part 3: Front End (Advanced)

https://github.com/annegolladay/api-lab

In this section, you’ll be making a front end that uses data from SWAPI. The goal is to be able to click a button and get all of the residents of the planet Alderaan listed out on the page.

Setup

1. Create a folder called “swapi” and three files inside - index.html, styles.css, and main.js
2. Open the folder up in VS Code
3. Run **npm init -y** which will create a package.json file
4. Install axios using npm

index.html

1. Create a basic HTML layout (doctype, html, head, body)
2. Connect the CSS file (using a link) and the JS file (script tag)
3. **Add another script tag, above the main.js script, to import axios**
    - since it’s in our node modules folder, the src of the script can just use the file path to get to axios, which is **“./node_modules/axios/dist/axios.min.js”**
3. In the body tag create a button that says “get residents” on it

main.js

1. Select the button using **querySelector** and save it to a variable
2. Write a function that just console logs a string like ‘button clicked’
3. Use **addEventListener** to attach the function you just wrote to a click event on the button
4. Open **index.html** in the browser (right click and copy path)
5. Click the button and check the console, if it’s working, move on to the next section

making a request

- As you complete this section, be sure to test along the way using Postman and console.logs

1. Now you’ll modify the function to make an axios call to SWAPI

2. Make an axios request that gets the information about the planet Alderaan (use the search query to request it, the how to on the search query is at the bottom of the Getting Started section of the documentation)
3. Inside the callback passed to the .then, loop over the residents array returned on the results. It's full of URLs.
4. In the loop, make another get request for each URL in the array.
5. You'll have another .then that has its own callback, inside which you should create an h2 element whose content is the name of the resident that you just requested. Append the h2 to your HTML document.

styles.css

- add any styles you'd like to your page

Submit

- Create a repo on GitHub and upload your text document (and swapi folder) to it