Javascript 1 Course Assignment Report

Introduction

Let me start off by saying thanks for an interesting and fun assignment! Initially when I opened the assignment, I must say I got a bit discouraged looking at the amount, but I decided to take a deep breath, and start doing one step at a time. Even if some of the “questions” put up a fair challenge, eventually I feel I did manage to solve the problems in an acceptable manner.

Main Body

Index Page

When I first looked at this task I thought it would be pretty straightforward, as I’ve already done this in a previous module assignment (or so I thought). I went about importing and converting the API the same way I did in my last module assignment, and everything looked alright.
I used my same method of approach for populating the HTML div with the info gathered from the JSON, not really encountering any problems.

Then came step 11 and my world fell apart (nearly). My first hurdle was, believe it or not, removing all the elements from the div (step 12). No matter what I did, everytime I clicked search, it refreshed the page/script, and repopulated the elements that got deleted.
After some time of reading and searching, I found the *event.preventDefault();*[1] method, and I thought that preventing the function from doing it’s default of reloading the script might work, and it certainly did.
Relieved of finally managing to solve that simple problem, and losing so much time doing it, I carried on. Next task was getting the input from the searchfield to return results from the cards to a new array and populate the page with the correct elements.
I knew initially that this certainly had to be done using the filter method, but did use a fair bit of time getting it to work correctly.
I did not want the search to be case specific, and I did want to return partial matches, as I feel a strict search-engine is not really a good search-engine.
I first had a variable for getting the searchinput, then another variable turning that search input to lowercase, but managed to bake the *.toLowerCase*[2] method into my filter function instead.
In the end what I did was making a function that filters the search-results based on the user input converted to lowercase, matched with the JSON object also converted to lowercase, and using the *.includes*[3] method to return partial matches. Adding the new elements to the HTML div with a suitable error message if no results was pretty straight forward.

[1] https://www.w3schools.com/jsref/event_preventdefault.asp
[2] https://www.w3schools.com/jsref/jsref_tolowercase.asp
[3] https://www.w3schools.com/jsref/jsref_includes_array.asp

Javascript 1 Course Assignment Report

Card Specific Page

After a rather “rough” start to the assignment, I was looking forward to the next task at hand, as a good challenge is a great way to learn.
Next up was the card specific page, and the task was to populate the page with the correct card from the “ID” given in the included script when clicking “View More”.
I started off by making a variable concatenating the API link and the ID, so it in return gave me a specific URL for the specific card API. I then fetched this link to get the JSON object, and then used this object to populate the required fields on the page.
I made an *if* statement checking if the result was undefined, and if it was it would return an error that the URL was invalid, if defined it would populate the page with the correct elements.

This was definitely not the hardest task, but all in all a fun exercise.

About Page

First task on the next page was replacing the word “Magic” with “Something”. At Least I hope that was the task, because I think there is a misspelling in the assignment. It says *“Replace the word “Magic” with the “Something” in all…”* I assumed that it was supposed to say *“with the word “Something”...”* and carried on with my task.
First thing I did was make a variable that returned the HTML element “aboutText”, and another one that returned the innerHTML content.
Using the *.replace*[4] method I first had an issue with it only replacing the first instance of the word Magic, ignoring the rest, but this was easily fixed by adding the /g to the word as a global modifier. I then removed the old text and replaced it with the modified one.

The next part was changing the style of the “moreInfoContent” div. Task was adding a toggle of that element that cycled “block” and “none”.
After creating the click event listener, I made a variable getting the element, thena simple if else to check what the style is. If it’s block, change to none and if it’s none, change to block.

[4] https://www.w3schools.com/jsref/jsref_replace.asp

Contact Form

The last task was validating the contact form, and displaying included error messages if failed.
I used some of my old work from a previous assignment here, as it was fairly similar, although the regex needed changing.
I used the page "https://regexr.com/"[5] to build my regex validators, as I feel it gives a good overview over what each part does, as regex can become a bit cluttered at times.
For checking that the name fields were not empty, I decided to use regex instead of actually checking if the input was empty by .length etc. I used *([^\s])* which uses () to make it into a group, [^] to match characters not in the set and /s to match whitespace.
Together they check that the input is not all whitespace, and will not return false if there is a space in the name, or even if the user by accident starts the name off with a space.

Next up was the email validation, which did not provide much of an issue, since I already had the regex I used for a previous assignment.
I used my regex *^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$* for this which uses ^ to mark the start of the string, *[]* to define the character sets with allowed chars inside, and +@ and +\. between the char sets to check that there is a @ and . present. The {2,4} makes sure that the domain name is between 2 and 4 chars long. $ defines the end of the string.

Now for the part that took the longest on this task, the phone number validation. It wasn't by any means an extreme challenge, but it took some work making the regex only allow for the 3 required inputs; *XXX XXX XXXX*, *XXX.XXX.XXXX* and *XXX-XXX-XXXX*.
What I managed to put together was this: *\d{3}(\s?|-?|\.?)\d{3}(\s?|-?|\.?)\d{4}*
I started by making the digit and length check, so I put in *\d{3}\d{3}\d{4}* as \d checks if it's a digit, and {3} would check that it's 3 in length etc.
I then had to make a check for either whitespace, "." or "-" in between them.
This I did by first making a capture group using () with my "arguments" inside. To check for either of the 3 arguments inside I used the "|" symbol which is for alternation, basically "or".
In plain letters the check would be: 555(whitespace or "-" or ".")555(whitespace or "-" or ".")5555.

With all the regex checks completed, checking and displaying was fairly straightforward.
I did an if/else statement for the error messages, so that if the user fixes the error, he would not have to reload the page to fix it, as the error is removed when pressing submit again.

[5] https://regexr.com

Javascript 1 Course Assignment Report

Conclusion

The assignment looked very daunting at first, and I had to take a deep breath before starting. But minus a few bumps and hurdles I feel I managed to solve the tasks well. My biggest challenge was the one in the index page where I had to first remove the elements, but the script was refreshing, and then the part to filter the results based on search input. The rest of the assignment was fairly straightforward and did not provide many issues, but I definitely feel I did learn a lot, especially about filtering and regex.

All in all I would say that this was a great assignment and I can't wait to go deeper into programming.

References

JavaScript References - *https://www.w3schools.com/*
RegEx editor - *https://regexr.com*
GIT repo - *https://github.com/*
GIT client - *https://www.gitkraken.com/*
Editor/IDE - *https://www.jetbrains.com/webstorm/* *(Thanks for the tip Cameron)*