WEB ENHANCEMENT REPORT

Ascent Physiotherapy - Tomio Walkley-Miyagawa

TABLE OF CONTENTS

WEBSITE BACKGROUND

Ascent Physiotherapy is a full service physiotherapy clinic located in Comox, BC. They offer physiotherapy, including manual and manipulative therapy, IMS, acupuncture and exercise prescription to the residents of the Comox Valley and surrounding area.

This is a redesign of the their existing website I created for DGL103.

FEATURES

- A quiz that matches symptoms with the ideal service
- View a list of services and an accompanying description
- Filter the list by symptom
- Add services to a invoice and view the total cost

IMPLIMENTATION

Because each of the three features involve a managing section of dynamic content I use the same pattern in each implementation, sharing functions when necessary. Each button is given a handler function, which updates relevant variables and then rebuilds the HTML for the section and outputs it to the DOM. The code for building the HTML is separated into its own function and returns an array of strings containing HTML markup. The returned value of that function is passed as an argument into the output function which appends the elements to the dynamic section. The markup is placed in an array of strings to allow for the output function to fade in each block sequentially.

The quiz works by granting points towards a service based on the likelihood of it being correct with the selected answer. Once the quiz is over, the service with the most points is displayed as the ideal service.

I used JSON for the quiz and services data. There is so much of it, it didn’t feel appropriate to include in the javascript as an object.

Techniques Used:

- DOM manipulation
- Event listeners
- Template literals
- Arrow functions
- Fetch API
- Array Methods

The W3C CSS Validation Service

W3C CSS Validator results for style.css (CSS level 3 + SVG)

Jump to: Warnings (97) Validated CSS

W3C CSS Validator results for style.css (CSS level 3 + SVG)

Congratulations! No Error Found.

This document validates as CSS level 3 + SVG !

To show your readers that you've taken the care to create an interoperable Web page, you may display this icon on any page that validates. Here is the XHTML you could use to add this icon to your Web page:

```
<p>
    <a href="http://jigsaw.w3.org/css-validator/check/referer">
        <img style="border:0;width:88px;height:31px"
            src="http://jigsaw.w3.org/css-validator/images/vcss"
            alt="Valid CSS!" />
    </a>
</p>
```

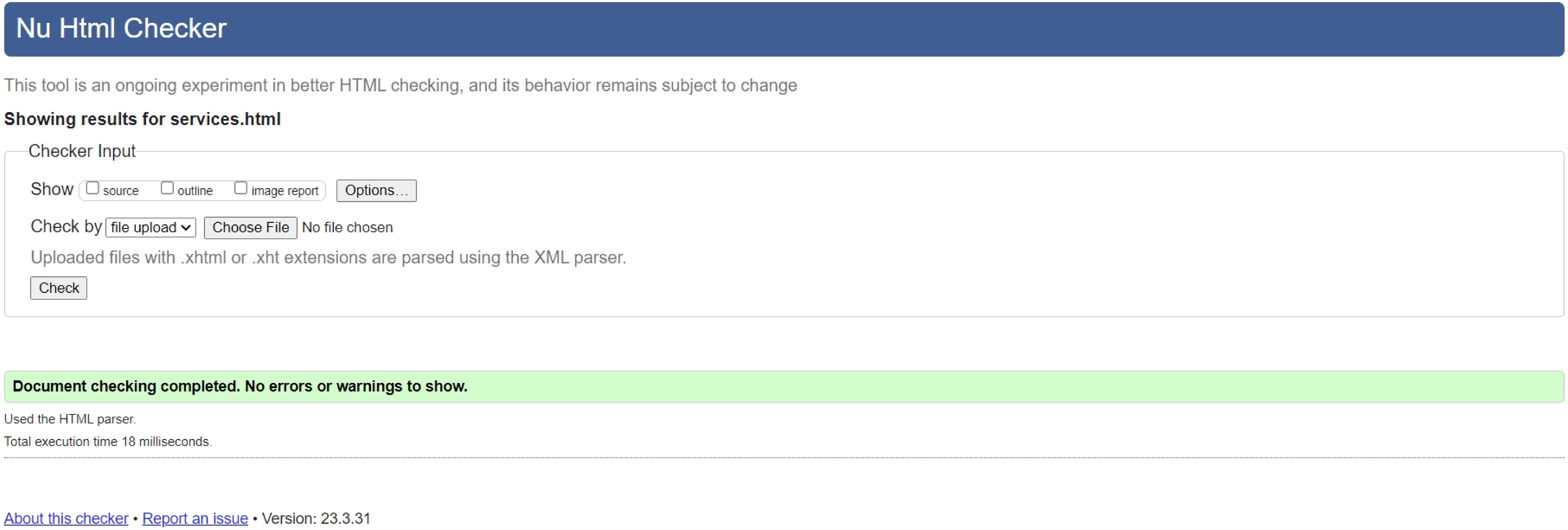
```
<p>
<a href="http://jigsaw.w3.org/css-validator/check/referer">
    <img style="border:0;width:88px;height:31px"
        src="http://jigsaw.w3.org/css-validator/images/vcss-blue"
        alt="Valid CSS!" />
    </a>
</p>
```

(close the img tag with > instead of /> if using HTML <= 4.01)

Interested in understanding what new technologies are coming out of W3C? Follow @w3cdevs on Twitter to keep track of what the future looks like!

Donate and help us build better tools for a better web.

If you like, you can download a copy of this image to keep in your local web directory, and change the XHTML fragment above to reference your local image rather than the one on this server.



Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for services.html

Checker Input

Show ☐ source ☐ outline ☐ image report Options…

Check by file upload Choose File No file chosen

Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

Check

Document checking completed. No errors or warnings to show.

Used the HTML parser.

Total execution time 18 milliseconds.

About this checker • Report an issue • Version: 23.3.31

BROWSER TESTING

- Windows 11/Chrome 111.0.5563.147
- Windows 11/Firefox 112.0
- MacOS Big Sur/Safari 16.3
- MacOS Big Sur/Safari 16.3

WORK EXPERIENCE

From the project work I learned the following:

I learned how to update the DOM and manipulate HTML elements using JavaScript. I further practiced my ability to write clean, maintainable and scalable code. I learned how to import JSON and parse a JSON string as a JavaScript object. I got practice in using arrays, iterating over them and transforming them in various ways. Lastly, I learned how to break down my code logical functions, making it easier to read, debug and scale.

I found the following helpful for completing the project:

- MDN web docs
- Grepper
  - Grepper is a browser extension that sources community answers to code search queries. It is embeded into the search results page so you don’t have to click into anything
- Stack Overflow
- Lectures (especially lectures on DOM interactions)

WORK EXPERIENCE CONTINUED

The difficulties I encountered with the project:

- Coming up with a standardized way to deal with dynamic content that was readable and reusable
- Developing robust code that accounted for edge cases.
  - Dealing with the state of the quiz UI and creating a system to allow you to go back to previous questions was difficult to implement elegantly considering some of the ramifications (ie. Are users previous answers preserved?)
- Filtering the service cards while the cards themselves are interactive objects with two states.
- Coming up with a method of keeping the state of the UI in sync with changing data that wasn't convoluted
- Preventing the code from becoming a long chain of functions calling other functions in which the functionality of the app is obfuscated
  - I tried to centralize the functionality around button click handler functions which call other functions abstracting away some of the code