AI-based Front-End Generation

WCAG Test.

Internship at iO Digital

Luuk Briels
467020
0.0.1

Internship Assignment Document

Contents.

Table of Contents.

𝓛.

Context.

1.1 Introduction

As one of the important aspects of the proof of concept. Generated pages should be as WCAG complaint as possible. WCAG or Web Content Accessibility Guidelines is a benchmark for website accessibility. Created by the World Wide Web Consortium (W3C), using WCAG guidelines is the best and the easiest way of making a website usable for all users. This is therefor also very important for the pages that the tool generates. This way, the pages can be properly viewed and used by all users.

1.2 Purpose of Test

To determine how well the generated pages using the proof of concept follow and integrate WCAG guidelines, I created this test. By doing these tests, I can identify which guidelines are not being followed correctly and propose possible solutions to solve these issues.

First I will talk more about the WCAG guidelines. Then I show how I tested the generated pages, checking each item on the list to see if the page meets the standards. If any issues are found, I will take detailed notes on what was wrong and why it didn't meet the guidelines. After finding the problems I will create possible solutions.

2.1 Guidelines

The WCAG guidelines consist of a set of rules related to accessibility which makes sure that content is more accessible to people. These guidelines are important for creating an accessible page. The WCAG guidelines are split into four main principles: Perceivable, Operable, Understandable, and Robust. Each of these principles has its own set of guidelines that need to followed to make the content accessible to all users.

The first principle is Perceivable. This is about making sure that users can perceive the information that is shown. This means that content must be shown in a way that users can see or hear, even if they have any disabilities. For example giving text alts for non text content like images and videos, makes sure that users who are visually impaired can still understand the content through screen readers.

The second principle is Operable. This is about making sure that users can navigate and use the interface. This includes making sure that all functionality is available from a keyboard because some users may not be able to use a mouse.

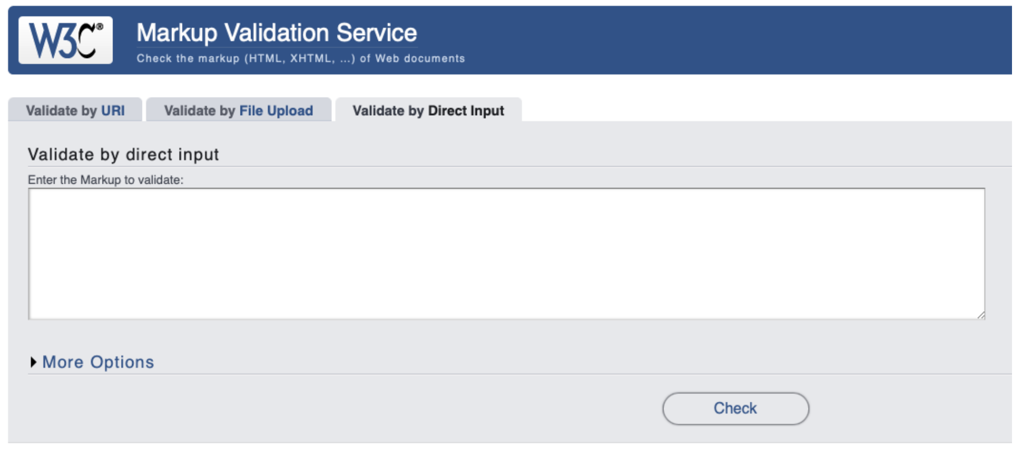
The third principle is Understandable. This is about making sure that users can understand the information as well as the operation of the user interface. Some examples of this are makign text readable and understandable and making sure that web pages appear and operate in predictable ways.

The fourth principle is Robust. This is about making sure that content can be used by a wide variety of browsers/user agents. This means that even when technologies changes, the content remains accessible.

Each of these principles is important in making the content accessible to all users. By following these guidelines the generated pages are both compliant with accessibility standards and also makes for a better user experience for everyone.

2.2 Tests

To test the generated pages for compliance with WCAG guidelines, I used a tool that analyzes the HTML and evaluates how well the pages meet the WCAG standards. This tool is officially provided by W3.org which is the organization responsible for developing and maintaining the web standards.


W3C
Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents
Validate by URI
Validate by File Upload
Validate by Direct Input
Validate by direct input
Enter the Markup to validate:
More Options
Check

The test is done by inputting the HTML code of the generated pages into the tool. This tool scans the HTML and gives a report on any accessibility issues it finds. It highlights areas where the pages do not meet the WCAG guidelines. This kind of analysis is important for making sure that the generated page is accessible to all users.

For the evaluation I did tests on three different page variants. The first variant was a page with no template but styled using CSS. The second variant was a page with no template but styled using Tailwind. The third variant was a page that included a template.

2.3 Results

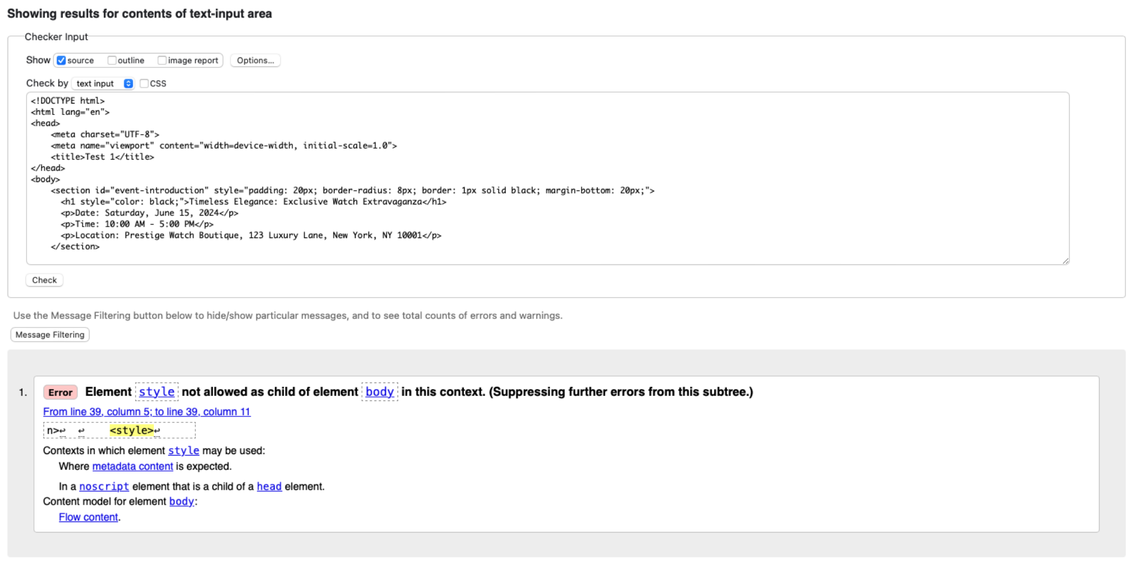
Test #1 - No template with CSS

The first test was done on three different generated pages that used no template and were styled using plain CSS. Running the test I found the same problem in all three pages. The results showed one guideline violation:

“Element style not allowed as child of element body in this context. (Suppressing further errors from this subtree.)”.

This error happened because the CSS was included inside the <body> tag of the HTML.

This issue happened because the AI generating the HTML is told to return only the content inside the <body> tag. Because of this it had no choice but to place the <style> tag within the <body> which causes the guideline violation.


Showing results for contents of text-input area

Checker Input

Show [✓ source] outline image report Options...

Check by text input CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Test 1</title>
</head>
<body>
    <section id="event-introduction" style="padding: 20px; border-radius: 8px; border: 1px solid black; margin-bottom: 20px;">
      <h1 style="color: black;">Timeless Elegance: Exclusive Watch Extravaganza</h1>
      <p>Date: Saturday, June 15, 2024</p>
      <p>Time: 10:00 AM - 5:00 PM</p>
      <p>Location: Prestige Watch Boutique, 123 Luxury Lane, New York, NY 10001</p>
    </section>
```

Check

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

Message Filtering

1. **Error Element style not allowed as child of element body in this context. (Suppressing further errors from this subtree.)**

From line 39, column 5; to line 39, column 11

```
n>↩   ↩      <style>↩
```

Contexts in which element style may be used:

Where metadata content is expected.

In a noscript element that is a child of a head element.

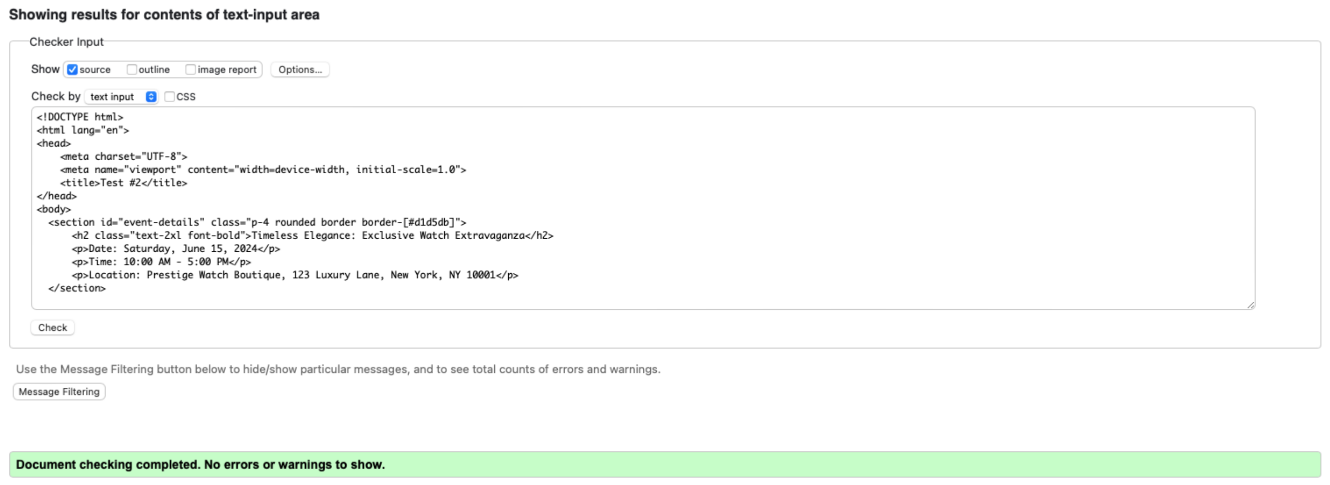
Content model for element body:

Flow content.

Test #2 - No template with Tailwind

The second test was done on three different generated pages that used no template and were styled using Tailwind CSS. Tailwind is a CSS framework that allows you to apply styles directly through class names added to HTML elements. I chose to test pages styled with Tailwind because it is a modern and popular way of doing styling in web development.

When I ran the test on the three generated pages it came back with no errors. The tool did not find any accessibility issues. This is because the CSS is done via classes instead of inside the <style>. So the AI did not use the <style> tag at all, which like the previous test would’ve caused an error if placed inside the <body>.


Showing results for contents of text-input area

Checker Input

Show [✓ source] outline image report Options...

Check by text input CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Test #2</title>
</head>
<body>
  <section id="event-details" class="p-4 rounded border border-[#d1d5db]">
      <h2 class="text-2xl font-bold">Timeless Elegance: Exclusive Watch Extravaganza</h2>
      <p>Date: Saturday, June 15, 2024</p>
      <p>Time: 10:00 AM - 5:00 PM</p>
      <p>Location: Prestige Watch Boutique, 123 Luxury Lane, New York, NY 10001</p>
  </section>
```

Check

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

Message Filtering

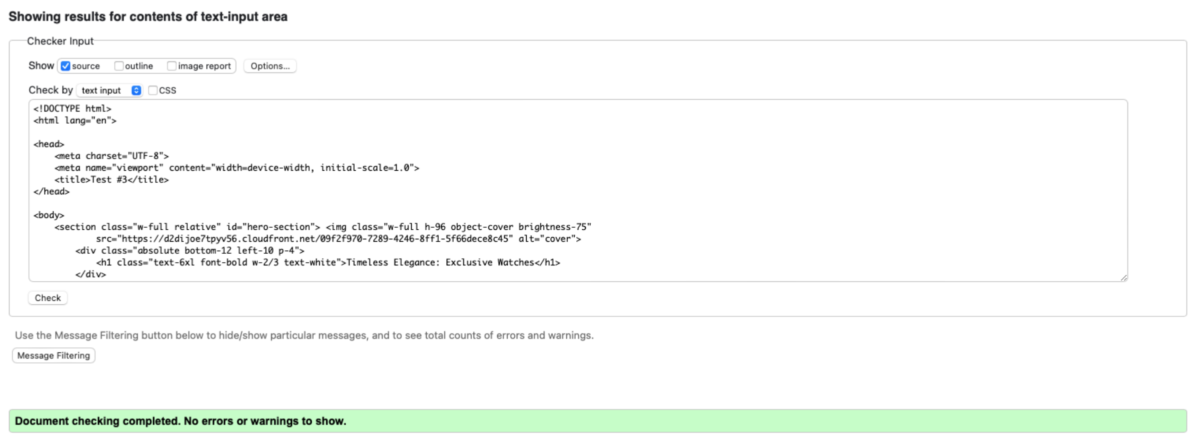
Document checking completed. No errors or warnings to show.

Test 3 - Template

The third and final test was done on three different generated pages that used a template. This test was important because I wanted to determine if the AI would make any changes to the template that could add WCAG compliance issues. Templates are first created by designers and developers so any changes made by the AI could potentially change the original design and create accessibility problems.

Before running the test I took a few steps to make sure the results are fair. First I reviewed the template itself to make sure that it was WCAG compliant. I checked all elements of the template by first checking it with the tool. This made sure that the template was compliant from the start in order to isolate any changes made by the AI during the content generation.

After generating the page I ran the accessibility test using the tool. The tool did not detect any errors which shows that the AI had successfully integrated the content into the template without making any changes that would change the WCAG compliance.



Showing results for contents of text-input area

Show [✓ source] outline image report

Check by text input CSS

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Test #3</title>
</head>

<body>
    <section class="w-full relative" id="hero-section"> <img class="w-full h-96 object-cover brightness-75"
            src="https://d2dijoe7tpyv56.cloudfront.net/09f2f970-7289-4246-8ff1-5f66dece8c45" alt="cover">
        <div class="absolute bottom-12 left-10 p-4">
            <h1 class="text-6xl font-bold w-2/3 text-white">Timeless Elegance: Exclusive Watches</h1>
        </div>
```

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

Document checking completed. No errors or warnings to show.

2.4 Improvements

Almost all test came back with no errors except one. The specific problem found was related to the placement of CSS within the HTML. In one of the tests the tool flagged an issue with the CSS being included inside the <body> tag. According to standard HTML practices and WCAG guidelines, style elements should be placed within the <head> section of the document. Including them inside the <body> is both considered improper and can also lead to different accessibility and rendering issues.

This problem happened because the AI is instructed to generate only the content inside the <body> tag. Because of this it had no choice but to place the <style> tag within the <body> which resulted in the guideline violation.

To fix this issue there are two solutions that can be implemented. The first one is to split the CSS into a separate file. By moving the CSS out of the HTML document and linking to an external stylesheet makes sure that the styles are applied correctly without violating any guidelines.

The second fix is to allow the AI to create the necessary tags outside of the <body>. By changing the AI's instructions to generate a complete HTML document including the <head> section, it can make sure that the CSS is placed in the correct location.

L.

Conclusion.

4 Conclusion

Through a series of tests on different page variants (one with no template styled using plain CSS, another with no template styled using Tailwind CSS, and a third using a pre-designed template), I got some good insights into the WCAG compliance of the generated pages. The test results showed that the pages styled with Tailwind and those using templates were compliant with WCAG standards. However the pages styled using plain CSS had an issue where the <style> tag was improperly placed inside the <body> tag which is against WCAG guidelines. However I mentioned two different solutions to this issue, which are relatively easy to implement.