

## Accesability & Web Components

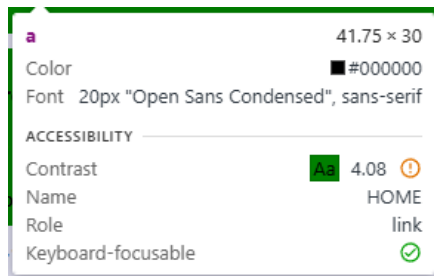
**(2) Color: Test the current color contrast (text/background), report the results of the test, and then fix them by changing the assigned colors.**

For this task I used the following contrast-checker from Web AIM:

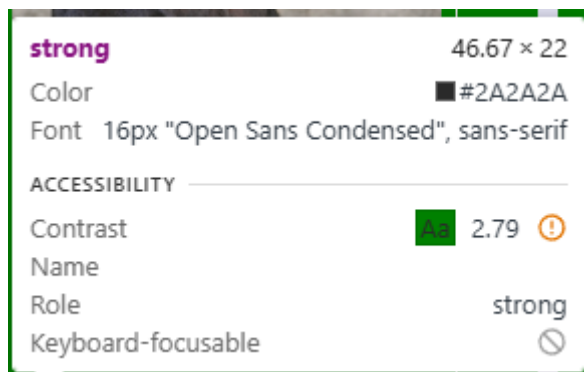
<https://webaim.org/resources/contrastchecker/>

I also used the Chrome-Browsers Contrast check to get fast result of the contrast ratio.

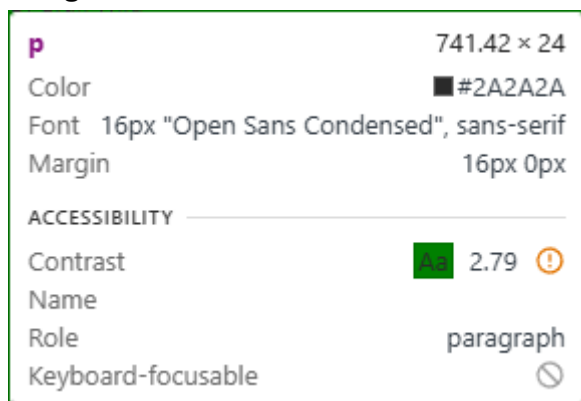
Navigation Links:



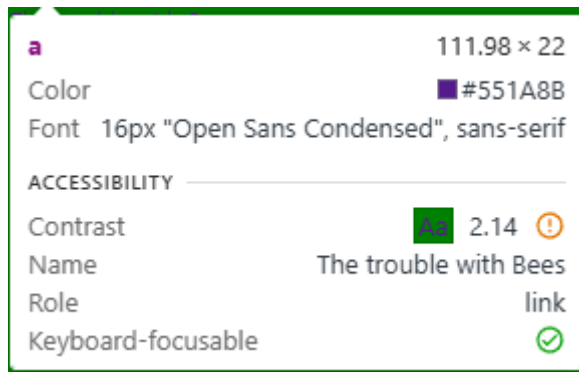
Name of the bears (More Bears Tab)



P Tags in the Articles:



## Related Tab Links:



For all contrast issues with normal written text I changed the text-color to black (#000000). Also I changed the background to more lighter green so the contrast ration also gets higher. For the Links in the “Related” Tab I changed the color of the links to: darkblue. With these changes the contrast issues should be solved.

(2) **Semantic HTML:** Report on what happens when you try to navigate the page using a screen reader. Fix those navigation issues.

For this Task i downloaded the NVDA Screen-Reader and tabbed through the webpage.

After that I went through the webpage using the screen Reader with TAB-Navigation.

I noted that:

- Not every text/content was accessible/readable with the Tab-Navigation e.g. Article-Text, Comments, “More Bears Tab” → Updated the Tab-index of the elements
- Language mix up. Aria – description in german, but website in english.

(2) **Audio:** The <audio> player isn't accessible to hearing impaired people — can you add some kind of accessible alternative for these users?

To transcribe this audio I added an div under the Audio file with a “p” Tag and the text from the audio. This way also people which cann’t not hear, get the information from the audio file.

(2) **Forms:**

- The <input> element in the search form at the top could do with a label, but we don't want to add a visible text label that would potentially spoil the design and isn't really needed by sighted users. Fix this issue by adding a label that is only accessible to screen readers.

For this task I created a class “sr-only” for screen reader only elements. Also deleted the hidden attribute, since it prevents screen readers.

```
<form class="search" role="search" aria-label="Search site">
  <label for="search" class="sr-only" >Search query</label>
  <input id="search" type="search" name="search" placeholder="Search article..." />
  <button type="submit">Go</button>
</form>
```

```
.sr-only{
  position: absolute;
  width: 1px;
  height: 1px;
  padding: 0;
  margin: -1px;
  overflow: hidden;
  clip: rect(0,0,0,0);
  white-space: nowrap; /* prevents it from expanding */
  border: 0;
}
```

- The two <input> elements in the comment form have visible text labels, but they are not unambiguously associated with their labels — how do you achieve this? Note that you'll need to update some of the CSS rule as well.

To Achieve this I added the label elements and associated them together with the “for” and “name” property.

(2) **Comment Section:** The show/hide comment control button is not currently keyboard-accessible. Can you make it keyboard accessible, both in terms of focusing it using the tab key, and activating it using the return key?

Already done before this exercise... I guess it was something like a div and I changed it to a button. A button is focusable and a div not (except it was set → Tabindex)

(4) **The table:** The data table is not currently very accessible — it is hard for screen reader users to associate data rows and columns together, and the table also has no kind of summary to make it clear what it shows. Can you add some features to your HTML to fix this problem?

Added aria-label with col-row description and the value.

Also added tabindex to cells to make cells focusable.

Current Solution: Works fine but not the best solution.

(6) Create a web component for the "Add comment" section. Use the shadow DOM and template syntax to encapsulate all related styles inside the component.

Nils Petsch

For this task I created a new file `comments-component.ts`. This holds the webcomponent.  
The template is on the bottom of the `index.html` file.