Noroff
School of technology and digital media

Project Exam 1

Taran Johannessen

Word count: 866

Design

After deciding on the topic I wanted a design that would feel thematically at home. The blog contains book reviews and recommendation. The design should be clean and inviting and inspire a calm and cozy feeling. The target audience for the blog is adults that enjoy reading and are proficient in the use of modern technology.

The home page has a hero image that represent a comfortable and relaxing atmosphere. It involves warm colours, and the image combined with the quote in the middle should help the user understand what the blog is about. The user is also met with images of book covers right away. The background colour for all pages is the colour of parchment. I think the overall colours scheme and the images on the about pages also help set the tone.

After implementing the styling from my prototype to the book reviews page I was uncertain how well it worked. In hindsight I think it would have worked better with more text than the large images. I could have included an extract from the copy and potentially the date it was posted. I knew I would need a lot of time to code all the functionality of the page, so I didn't prioritize this change. It is something that I would change if I had had more time.

One thing I would do differently next time is spend more time in the design phase to properly layout different screen sizes. I made a prototype for mobile and desktop view, but ended up using a lot of time on coding to find something that worked well for the medium sized screens.

Technical

I had a detailed plan for how I would spend the time with the coding, but quickly went out of order with the different issues I tackled. Next time I will allow for more flexibility in my plan.

The HTML and CSS coding went well. I did not run into issues before I started implementing JavaScript. I had no clear idea of how to do any of the major tasks: the carousel, image modal and the "view more" for the blog list page. My JavaScript skills are beginner level and

the only code I felt confident in doing was the form validation and the fetching of the API. I relied heavily on tutorial videos to complete the rest of the code. All of them are linked below.

One of the biggest issues I faced was how to make the arrows on the carousel work. I followed a video guide (CodingNepal, 2023) that seemed to have exactly what I was going for, but on my carousel the right arrow would disappear permanently after clicking it. I did a lot of troubleshooting by googling and searching through websites about web development, but I was not able to find a solution. In the end I solved the problem with the help of ChatGPT which I understand is not ideal for a learning process. I included the chatlog in my references so I can be as transparent as possible.

A lot of the guides I followed (DotWebdesign, 2022 & Mr. Web Desinger, 2021) was difficult to implement because I was fetching my elements from an API while the guide had it hardcoded. Some of my functions ended up being very long and difficult for me to read (like the getDetails function in blog-post.js, because everything needed to be done in the same place the API was fetched. I am not confident this is the best way to solve it but with my limited skills this was what I was able to do.

When it comes to making the code readable and structured, I think I have a long way to go. This is a place where I'll need to trust my experience and growth over time.

WCAG guidelines, content management and SEO

I used WAVE evaluation tool and a colourblind web page filter from Toptal.com to make sure the site follows the WCAG guidelines.

There were some issues connected to WordPress. Setting it up and hosting a domain for the API went well, but figuring out how to get certain elements from the API was difficult. I tried using both regular posts and the WooCommerce plug in for this, but I found neither of them optimal. With the help of the REST API Handbook I was able to make it work, but the way I am getting information about book title and author is not ideal. I added the information as

tags and categories and there is most likely a better way to do it, maybe with the use of plugins. The REST API Handbook helped me figure out how to get the URL to fetch more than 10 posts and how to get embedded information.

My focus when it comes to SEO was to create high-quality, relevant, and engaging content. The webpage has a simple site structure and is optimized for mobile users. All pages have relevant titles and meta descriptions. The individual blogposts are not optimized for search engines as they include no keywords or categories. Next time I will do even more research beforehand, to find out what keywords are relevant.

References

CodingNepal. (2023, Sep 2). *Create Responsive Image Slider in HTML CSS and JavaScript | Image Slider HTML CSS & JavaScript. YouTube*. https://youtu.be/PsNaoDhzQm0?si=vSweLdF0cBucdLUq

DotWebdesign. (2022, Jul 6). *Load More Button Javascript.* YouTube. https://youtu.be/rMeN2_pXGto?si=t-Br6cq3_mEcyJ5_

Mr. Web Designer. (2021, Jul 20). *Create A Responsive Popup Image Gallery Using HTML CSS And Vanilla Javascript*. YouTube. https://www.youtube.com/watch?v=QghhoJBdw7A

OpenAI. (2023). *ChatGPT*. https://chat.openai.com/ *(ChatGPT was used to make the copy)*

Purkiss, R. *multi image modal - CSS and JS*. Codepen. https://codepen.io/r3dg3cko/pen/ZLryQG

Toptal. *Colorblind Web Page Filter*. https://www.toptal.com/designers/colorfilter

Web Dev Simplified. (2019, Apr 20). *Build a Popup With JavaScript*. YouTube. https://youtu.be/MBaw_6cPmAw?si=Fc0wseXJh54P4cIP

WordPress. *Rest API Handbook*(version 5.6). https://developer.wordpress.org/rest-api/

ChatGPT chatlog

You
can you explain this javascript: const initSlider = () => { const reviewList = document.querySelector(".carousel"); const slideButtons = document.querySelectorAll(".slide-button"); const maxScrollLeft = reviewList.scrollWidth - reviewList.clientWidth; // slide through posts according to the slide button clicks slideButtons.forEach((button) => { button.addEventListener("click", () => { const direction = button.id === "prev-slide" ? -1 : 1; const scrollAmount = reviewList.clientWidth * direction; reviewList.scrollBy({ left: scrollAmount, behavior: "smooth" }); }); }); const handleSlideButtons = () => { slideButtons[0].style.display = reviewList.scrollLeft <= 0 ? "none" : "block"; slideButtons[1].style.display = reviewList.scrollLeft >= maxScrollLeft ? "none" : "block"; }; reviewList.addEventListener("scroll", () => { handleSlideButtons(); }); }; window.addEventListener("load", initSlider);

ChatGPT
ChatGPT
Sure, this JavaScript code is for creating a slider functionality for a carousel. Let's break it down step by step:

- **Initialization Function** (`initSlider`):
- It's a function declared using arrow syntax (`const initSlider = () => { ... }`).
- It starts by selecting elements from the DOM:
- `reviewList`: Represents the carousel element with the class name "carousel".
- `slideButtons`: Represents the buttons to navigate the carousel (presumably for moving left and right).
- `maxScrollLeft`: Calculates the maximum scrollable distance within the carousel.
- **Event Listeners**:
- For each `slideButton` (presumably the left and right navigation buttons), it sets up a click event listener.
- When a button is clicked, it determines the direction based on the button's ID (`prev-slide` or `next-slide`), calculates the amount to scroll, and scrolls the `reviewList` accordingly using `scrollBy` with a smooth behavior.
- `handleSlideButtons` **Function**:
- Updates the display of the navigation buttons based on the current scroll position.
- The first button (`slideButtons[0]`) is displayed (`"block"`) if the scroll position is not at the beginning (`scrollLeft <= 0`), otherwise, it's hidden (`"none"`).
- The second button (`slideButtons[1]`) is displayed if the scroll position is not at the maximum scroll distance (`scrollLeft >= maxScrollLeft`), otherwise, it's hidden.
- **Event Listener for Scroll**:
- Whenever the `reviewList` element is scrolled, the `handleSlideButtons` function is called to update the visibility of the navigation buttons based on the scroll position.
- **Initialization on Window Load**:
- When the window is loaded, it triggers the `initSlider` function, initializing the slider functionality.

This code creates a carousel/slider with left and right navigation buttons that enable smooth scrolling through the content. The buttons' visibility is dynamically updated based on the scroll position within the carousel.

You
slideButtons[1].style.display = reviewList.scrollLeft >= maxScrollLeft ? "none" : "block"; this part does not work as intended and the button becomes invisible as soon as it is clicked and does not become visible again when scrolled to the left. how do I fix it?
ChatGPT

ChatGPT
It seems like there might be an issue with how the **maxScrollLeft** value is being calculated or updated in the code, leading to incorrect comparison logic for displaying the navigation button.
To fix the problem and ensure that the right navigation button becomes visible again when scrolled to the left, you can modify the logic inside the **handleSlideButtons** function:
javascriptCopy code

```
const handleSlideButtons
                0                                                   0    "none"
"block"
                1
                                                "block"    "none"
```

This change ensures that the right navigation button (**slideButtons[1]**) becomes visible (**"block"**) when the **reviewList.scrollLeft** position is less than the maximum scrollable distance (**maxScrollLeft**). This should fix the issue you described, allowing the right button to reappear when there's content to scroll to the right.

You
the new code still makes the button become invisible as soon as it is clicked

ChatGPT
ChatGPT
Apologies for the confusion earlier. It seems like the problem might be related to the way the **maxScrollLeft** value is calculated or updated. Let's try an alternative approach:
Instead of calculating **maxScrollLeft** once during initialization, let's dynamically calculate it within the **handleSlideButtons** function. This should accurately determine the maximum scrollable distance each time the scroll event occurs:
javascriptCopy code

```
const handleSlideButtons
                0                                                   0    "none"
"block"


  const

                1
                                                "block"    "none"
```

By recalculating **maxScrollLeft** inside the **handleSlideButtons** function, it should accurately reflect the maximum scrollable distance, allowing the right navigation button to appear when there's more content to scroll to the right.