A screenshot of a cell phone

Description automatically generated

# Project Exam 1

Jean Larroza Arcenal

Front-end Development

May 25, 2023

Word count

Summary: 255 / Main text: 620

## Table of Contents

[1. Summary………………………………………………………………………………………………………………………3](#_Toc9086)

[2. Body……………………………………………………………………………………………………………………………..4](#_Toc9087)

[2.1. Introduction…………………………………………………………………………………………………………..4](#_Toc9088)

[2.2. Main section of report…………………………………………………………………………………..……](#_Toc9089)5-6

[2.3. Conclusion………………………………………………………………………………………………………..……6](#_Toc9090)

3. References ……………………………………………………………………………………………………………………7

## Summary

The process of creating a blog site using WordPress as a content management system simplifies the process of creating and managing a blog site.

I successfully fetched the latest posts from a WordPress website using its REST API and processed the data to create a carousel on my website homepage.

I utilized various variables and functions to handle the carousel functionality, including selecting the carousel element from the DOM, constructing HTML strings for each post, updating slide positions, and managing button states.

The helpers.js file provides additional functionality for scrolling, back-to-top button, and mobile menu interactions, including smooth scrolling behavior.

In terms of refactoring, just made changes to endpoint variables on different pages and exported a fetchingData function from the utility.js file, which can be used to retrieve data from the server asynchronously.

Installing WordPress as a headless CMS on my web host and configuring it accordingly provides an intuitive and user-friendly interface.

On my responsive, when a media query minimum width is 768 pixels, the burger menu hides, and at 767 pixels the burger menu displays.

When the screen is 575 pixels or larger, the slides button on the carousel content displays, and the screen of 574 pixels and below hides the slides button.

However, I encountered an issue when integrating Hotjar for feedback, where the feedback button took a long time to display and disappeared upon the index page refresh. Exploring alternative feedback tools or investigating potential conflicts between their code and the feedback application in the future maybe can solve an issue.

1. **Body**

**2.1 Introduction**

The project design for my blog site went smoothly, and I successfully created it using WordPress. To retrieve the latest posts from the WordPress website's REST API, I utilized the latestPostEndpoint variable, which is a string representing the endpoint URL. This URL includes query parameters to specify the number of posts to retrieve per page (set to 5) and the sorting order by date.

To extract the relevant properties from each post object, such as id, title, \_embedded, and excerpt, I used destructuring.

To display the posts in a carousel format, I implemented a flex-based carousel element that only appears if there is at least one post in the mapData array.

Next, I added event listeners to the next and prev buttons, which triggered the executeCarousel function when clicked.

In addition, I utilized a helpers.js file to set up event listeners for scrolling, clicking the back-to-top button, and interacting with a mobile menu.

For responsive design, I implemented media queries to adjust the appearance of certain elements based on the screen size.

To retrieve data from the server asynchronously, I imported a function named fetchingData from the utility.js file. This function made an asynchronous HTTP GET request to a specified endpoint when called with the desired endpoint as an argument.

In terms of hosting, I installed WordPress on my web host, one.com, as a headless CMS.

**2.2 Main section of the report**

**Project Design**

The project design went well.

I created my first blog site using a post from WordPress, a widely-used content management system (CMS).

The latestPostEndpoint variable is a string that represents the endpoint URL to fetch the latest posts from a WordPress website's REST API. It includes query parameters to specify the number of posts to retrieve per page which is equal to 5 and the sorting order by date.

Each post object is destructured to extract the relevant properties such as id, title, \_embedded, and excerpt. These properties are then used to create a new object data that contains the required data for each post.

The carousel element displays flex if there is at least one post in the mapData array. The map data array is iterated using forEach. For each post, an HTML string item data is constructed using template literals, incorporating the post's title, featured image URL, excerpt, and a link to view details. This HTML string is appended to the carousel element's innerHTML, effectively adding the post as a new slide to the carousel.

Event listeners are added to the next and prev buttons, triggering the executeCarousel function when clicked. The executeCarousel function adjusts the translateValue and currentIndex based on the clicked action and updates the slide positions accordingly.

The checkButtonStatus function enables or disables the next and prev buttons based on the current slide's index. If the current slide is the last one, the next button is disabled. If the current slide is the first one, the prev button is disabled.

The function getPostDetails selects various elements from the HTML document using getElementById and querySelector. It sets up event listeners to click events on the document and a close button. When a click event occurs on the document, it checks if the target is the modal element. If so, it hides the modal by adjusting its opacity and display properties. When the close button is clicked, it also hides the modal by adjusting the opacity and display properties. It sets up click event listeners for each image. When an image is clicked, it updates the source and alt attributes of the image inside the modal with the clicked image's attributes. It then displays the modal and gradually increases its opacity.

The functionality displays a modal dialog box with an image when an image is clicked and allows the user to close the modal by clicking outside the content area or on a close button.

A helpers.js file is to set up event listeners for scrolling, clicking the back-to-top button, and interacting with a mobile menu.

Inside the event listener function, there is an if-else statement that checks the value of window.pageYOffset, which represents the number of pixels the document has been scrolled vertically. If the scroll position is greater than 100 pixels, display the backToTopBtn element to flex, making it visible. Otherwise, if the scroll position is less than or equal to 100 pixels, display to 'none', hiding the button.

 On my responsive, when a media query minimum width is 768 pixels, the burger menu hides, and at 767 pixels the burger menu displays.

When the screen is 575 pixels or larger, the slides (carousel) button displays, and the screen of 574 pixels and below hides the slides button.

 The imported function on utilility.js named fetchingData is responsible for making an asynchronous HTTP GET request to a specifiedendpoint. By calling the fetchingData function with the desired endpoint, it can retrieve data from the server using an asynchronous approach.

I installed WordPress on my web host one.com as a headless CMS. On filemanager.one.com/config, I added my domain name to make a CMS headless.

Despite the successful implementation, I encountered a challenge during the integration of Hotjar for feedback. The feedback button experienced delays in displaying and disappeared upon refreshing the index page.

To resolve this issue, I plan to explore alternative feedback tools and investigate potential conflicts between the code and the feedback application.

By doing so, I aim to address this concern and improve the feedback functionality of the website in the future.

**2.3 Conclusion**

The project design for my blog site using WordPress as a headless CMS was successful. I was able to fetch the latest posts from the WordPress REST API and display them in a carousel format on the website. The implementation involved utilizing the latestPostEndpoint variable, destructuring post objects, and creating a new data object with the necessary post information.

The responsive design aspect of the website ensured optimal user experience across different screen sizes. Additionally, event listeners were set up for scrolling, back-to-top button functionality, and mobile menu interaction using the helpers.js file.

To retrieve data asynchronously from the server, the fetchingData function from the utility.js file was imported.

Overall, the project design and implementation of the blog site using WordPress and additional functionalities were accomplished effectively.

## References

* <https://www.w3schools.com/howto/howto_css_center-vertical.asp>
* https://css-tricks.com/almanac/properties/l/line-clamp/
* <https://www.w3schools.com/howto/howto_js_slideshow.asp>
* https://www.freecodecamp.org/news/rest-api-best-practices-rest-endpoint-design-examples/
* https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener