Hashim Aziz Muhammad

Task 13 - JSON and Client-Server Communication

Course: JavaScript Programming

Reviewed By: Sam Baloyi

Date Submitted: April 2, 2024, 8:04 a.m.

Date Reviewed: April 3, 2024, 9:16 a.m.

Student number: HA24010012783

Scores

Completeness: 4/4

Efficiency: 4/4

Style: 4/4

Documentation: 4/4

Positive aspects of the submission

Hi Hashim. I hope you are doing well. Thank you for submitting your work.

Hashim, firstly, your HTML structure is well-organised and straightforward. The division into sections for income, expenses, and totals makes the content clear and easy to understand for users. The use of semantic HTML elements enhances accessibility and SEO.

Your JavaScript code effectively creates income and expense objects using constructor functions, which is a suitable approach for organising and managing data. The use of session storage to store income and expense data allows for persistent storage across page reloads, ensuring a seamless user experience.

The implementation of the addIncome and addExpense functions is commendable. These functions prompt users for input to create new income and expense objects, respectively, and update the session storage and UI accordingly. The use of prompt boxes for user interaction is appropriate and provides a simple way for users to input data.

Your code for updating the income and expense lists in the HTML is well-written and efficient. The updateIncomeList and updateExpenseList functions iterate over the income and expense data retrieved from session storage, respectively, and dynamically create list items to display the information. This approach ensures that the UI reflects the current state of income and expense data accurately.

The calculateTotals function accurately calculates the total income and expense amounts based on the data stored in session storage. The totals are then displayed in the HTML, providing users with a clear overview of their financial situation.

Finally, the calculateDisposable function calculates the disposable income by subtracting total expenses from total income. It then prompts users to input the amount they would like to save and updates the UI accordingly. The implementation handles edge cases, such as invalid input or attempting to save more than the disposable income, effectively.

Overall feedback

In conclusion, Hashim. your submission demonstrates a solid understanding of JSON and client-server communication concepts. Your code is well-structured, readable, and effectively implements the required functionality. Keep up the excellent work.