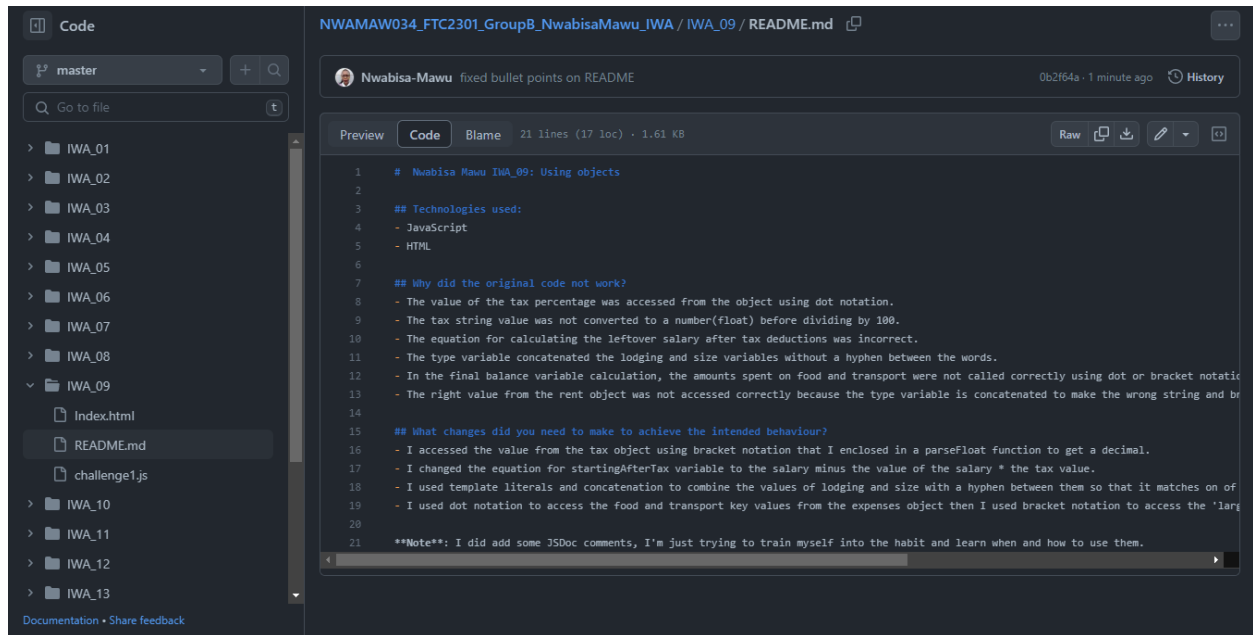


DWA_03.4 Knowledge Check_DWA3.1

1. Please show how you applied a Markdown File to a piece of your code.

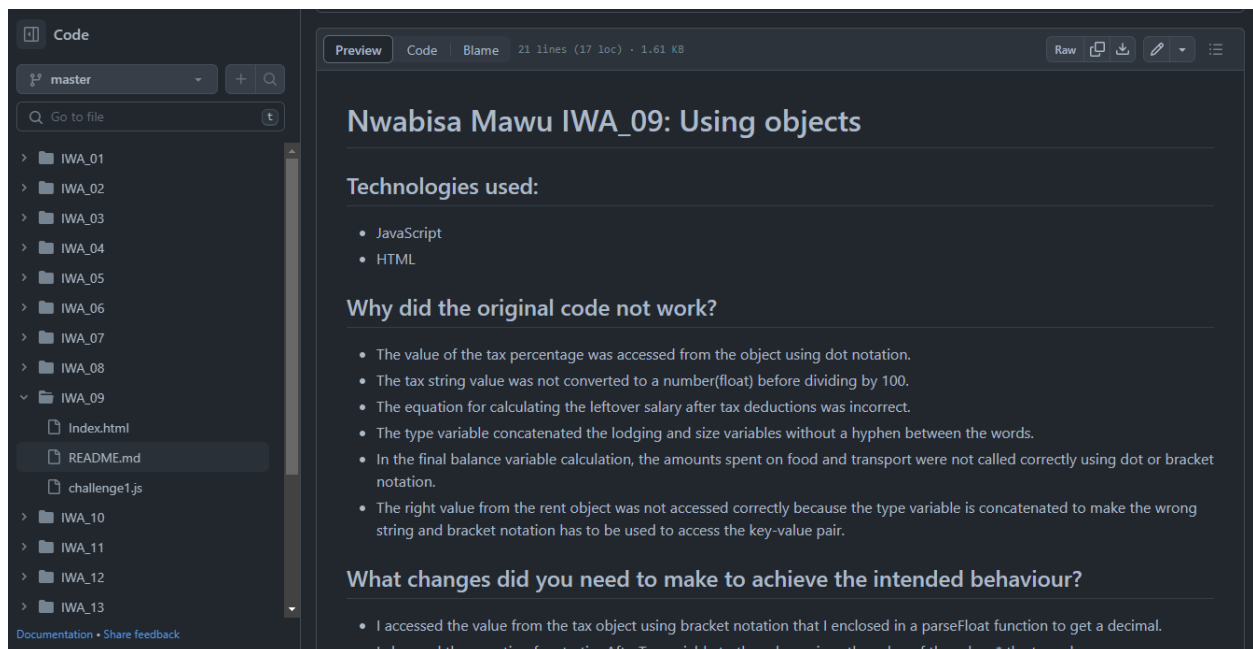
This is the code used to make the title headings and the questions.



The screenshot shows a code editor interface. On the left is a file explorer with a tree view containing folders IWA_01 through IWA_13 and files Index.html, README.md, and challenge1.js. The main editor area displays a file named 'NWAMAW034_FTC2301_GroupB_NwabisaMawu_IWA / IWA_09 / README.md'. The code is written in Markdown and includes a title, technologies used, a section on why original code failed, and a section on changes made to achieve the intended behavior. The code is as follows:

```
1 # Nwabisa Mawu IWA_09: Using objects
2
3 ## Technologies used:
4 - JavaScript
5 - HTML
6
7 ## Why did the original code not work?
8 - The value of the tax percentage was accessed from the object using dot notation.
9 - The tax string value was not converted to a number(float) before dividing by 100.
10 - The equation for calculating the leftover salary after tax deductions was incorrect.
11 - The type variable concatenated the lodging and size variables without a hyphen between the words.
12 - In the final balance variable calculation, the amounts spent on food and transport were not called correctly using dot or bracket notation.
13 - The right value from the rent object was not accessed correctly because the type variable is concatenated to make the wrong string and bracket notation.
14
15 ## What changes did you need to make to achieve the intended behaviour?
16 - I accessed the value from the tax object using bracket notation that I enclosed in a parseFloat function to get a decimal.
17 - I changed the equation for startingAfterTax variable to the salary minus the value of the salary * the tax value.
18 - I used template literals and concatenation to combine the values of lodging and size with a hyphen between them so that it matches on of
19 - I used dot notation to access the food and transport key values from the expenses object then I used bracket notation to access the 'largest'
20
21 **Note**: I did add some JSDoc comments, I'm just trying to train myself into the habit and learn when and how to use them.
```

This is the preview of the readme file with the code above.



The screenshot shows the same code editor interface, but the main editor area now displays the rendered preview of the Markdown file. The preview includes the title 'Nwabisa Mawu IWA_09: Using objects', the technologies used (JavaScript and HTML), the section 'Why did the original code not work?' with a list of five bullet points, and the section 'What changes did you need to make to achieve the intended behaviour?' with a list of three bullet points. The preview is as follows:

Nwabisa Mawu IWA_09: Using objects

Technologies used:

- JavaScript
- HTML

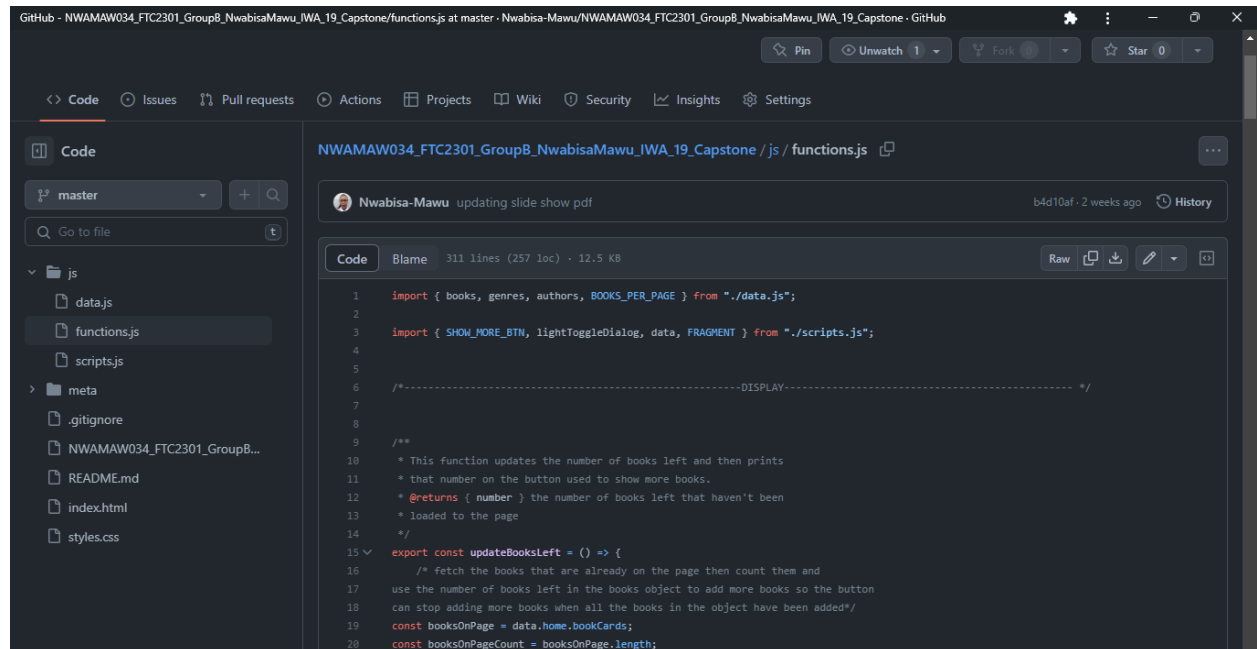
Why did the original code not work?

- The value of the tax percentage was accessed from the object using dot notation.
- The tax string value was not converted to a number(float) before dividing by 100.
- The equation for calculating the leftover salary after tax deductions was incorrect.
- The type variable concatenated the lodging and size variables without a hyphen between the words.
- In the final balance variable calculation, the amounts spent on food and transport were not called correctly using dot or bracket notation.
- The right value from the rent object was not accessed correctly because the type variable is concatenated to make the wrong string and bracket notation has to be used to access the key-value pair.

What changes did you need to make to achieve the intended behaviour?

- I accessed the value from the tax object using bracket notation that I enclosed in a parseFloat function to get a decimal.
- I changed the equation for startingAfterTax variable to the salary minus the value of the salary * the tax value.

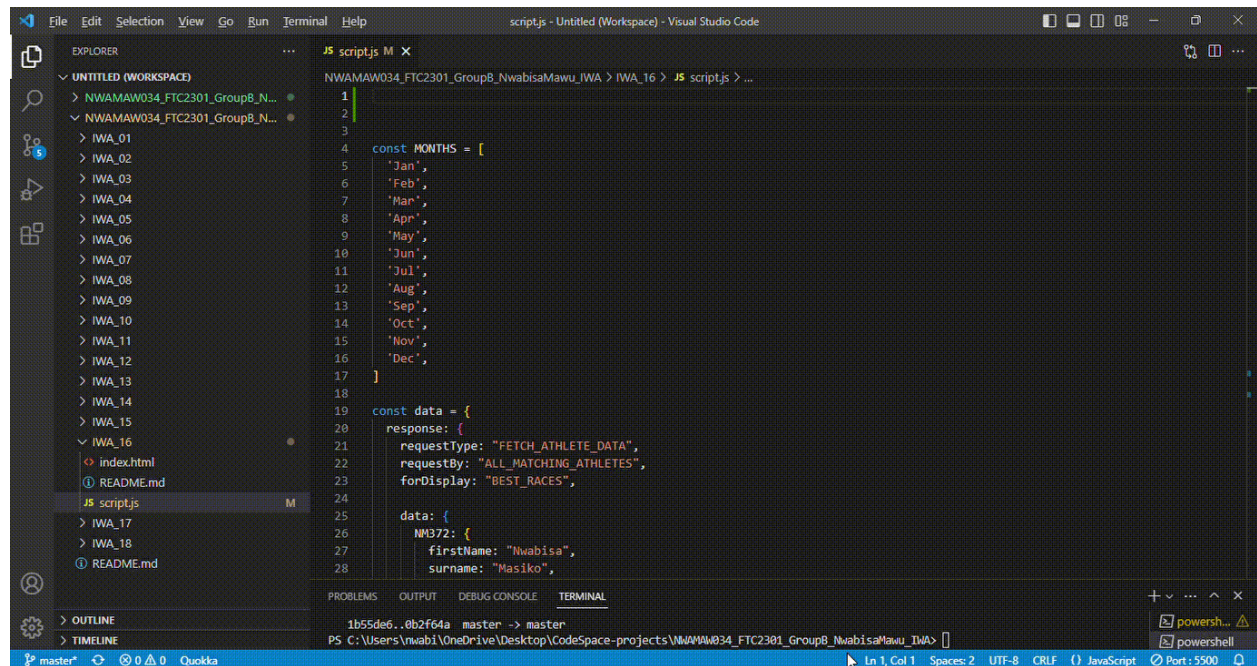
2. Please show how you applied JSDoc Comments to a piece of your code.



The screenshot shows a GitHub repository for 'NWAMAW034_FTC2301_GroupB_NwabisaMawu_IWA_19_Capstone'. The file 'functions.js' is open, showing a JSDoc comment for the 'updateBooksLeft' function. The comment describes the function's purpose, parameters, and return value.

```
1 import { books, genres, authors, BOOKS_PER_PAGE } from "../data.js";
2
3 import { SHOW_MORE_BTN, lightToggleDialog, data, FRAGMENT } from "../scripts.js";
4
5
6 /*-----DISPLAY-----*/
7
8 /**
9  * This function updates the number of books left and then prints
10  * that number on the button used to show more books.
11  * @returns { number } the number of books left that haven't been
12  * loaded to the page
13  */
14
15 export const updateBooksLeft = () => {
16   /* fetch the books that are already on the page then count them and
17    use the number of books left in the books object to add more books so the button
18    can stop adding more books when all the books in the object have been added*/
19   const booksOnPage = data.home.bookCards;
20   const booksOnPageCount = booksOnPage.length;
```

3. Please show how you applied the @ts-check annotation to a piece of your code.

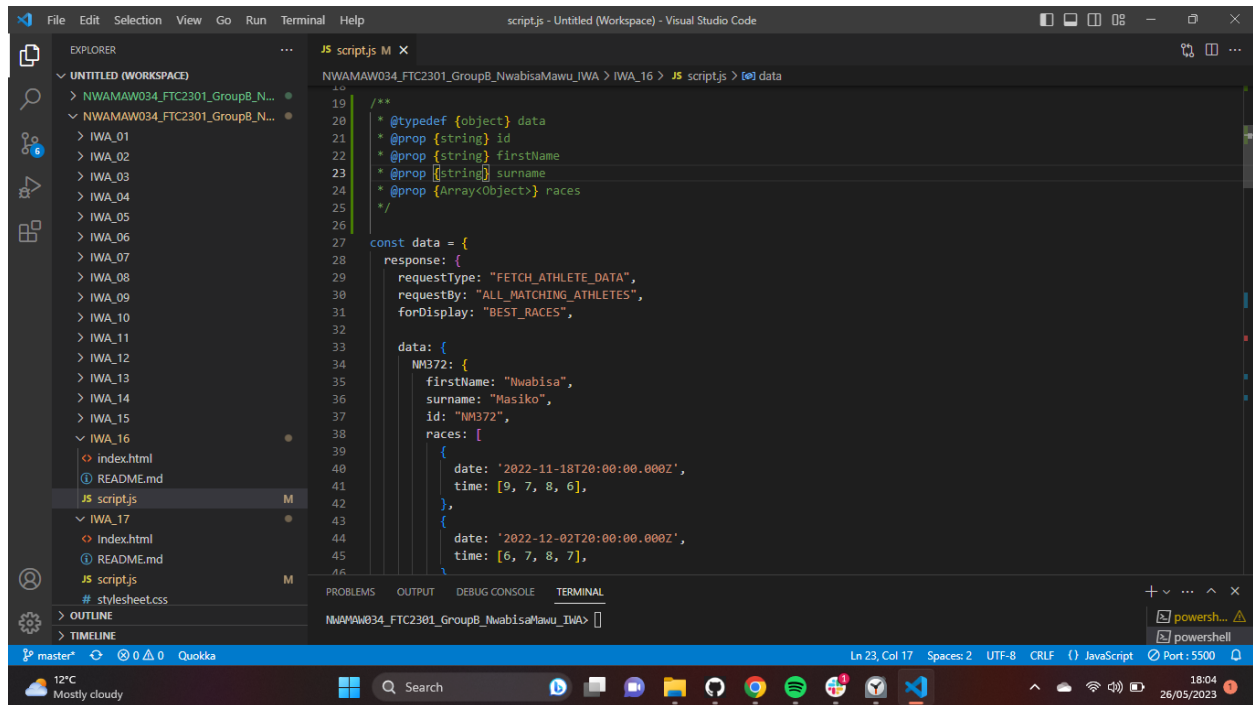


The screenshot shows a Visual Studio Code editor with a file named 'script.js' open. The file contains a TypeScript file with a `@ts-check` annotation at the top. The code defines a `MONTHS` array and a `data` object.

```
1
2
3
4 const MONTHS = [
5   'Jan',
6   'Feb',
7   'Mar',
8   'Apr',
9   'May',
10  'Jun',
11  'Jul',
12  'Aug',
13  'Sep',
14  'Oct',
15  'Nov',
16  'Dec',
17 ]
18
19 const data = {
20   responseType: "FETCH_ATHLETE_DATA",
21   requestBy: "ALL_MATCHING_ATHLETES",
22   forDisplay: "BEST_RACES",
23
24   data: {
25     NB372: {
26       firstName: "Nwabisa",
27       surname: "Masiko",
28     }
```

The parameter types in the {} had a * before them so when the type check is activated with @ts-check, the type declarations inside the brackets were highlighted and had to be fixed.

4. As a BONUS, please show how you applied any other concept covered in the 'Documentation' module.



The screenshot shows the Visual Studio Code editor with a workspace named 'scriptjs - Untitled (Workspace)'. The Explorer panel on the left shows a project structure with folders 'IWA_01' through 'IWA_17' and files 'index.html', 'README.md', and 'scriptjs'. The main editor displays a TypeScript file named 'scriptjs M X' with the following code:

```
19  /**
20   * @typedef {object} data
21   * @prop {string} id
22   * @prop {string} firstName
23   * @prop {string} surname
24   * @prop {Array<Object>} races
25   */
26
27  const data = {
28    response: {
29      requestType: "FETCH_ATHLETE_DATA",
30      requestBy: "ALL_MATCHING_ATHLETES",
31      forDisplay: "BEST_RACES",
32
33      data: {
34        NM372: {
35          firstName: "Nwabisa",
36          surname: "Masiko",
37          id: "NM372",
38          races: [
39            {
40              date: '2022-11-18T20:00:00.000Z',
41              time: [9, 7, 8, 6],
42            },
43            {
44              date: '2022-12-02T20:00:00.000Z',
45              time: [6, 7, 8, 7],
46            }
47          ]
48        }
49      }
50    }
51  }
```

The bottom status bar shows 'Ln 23, Col 17', 'Spaces: 2', 'UTF-8', 'CRLF', 'JavaScript', and 'Port: 5500'. The system tray at the bottom indicates '12°C Mostly cloudy' and the date '26/05/2023'.
