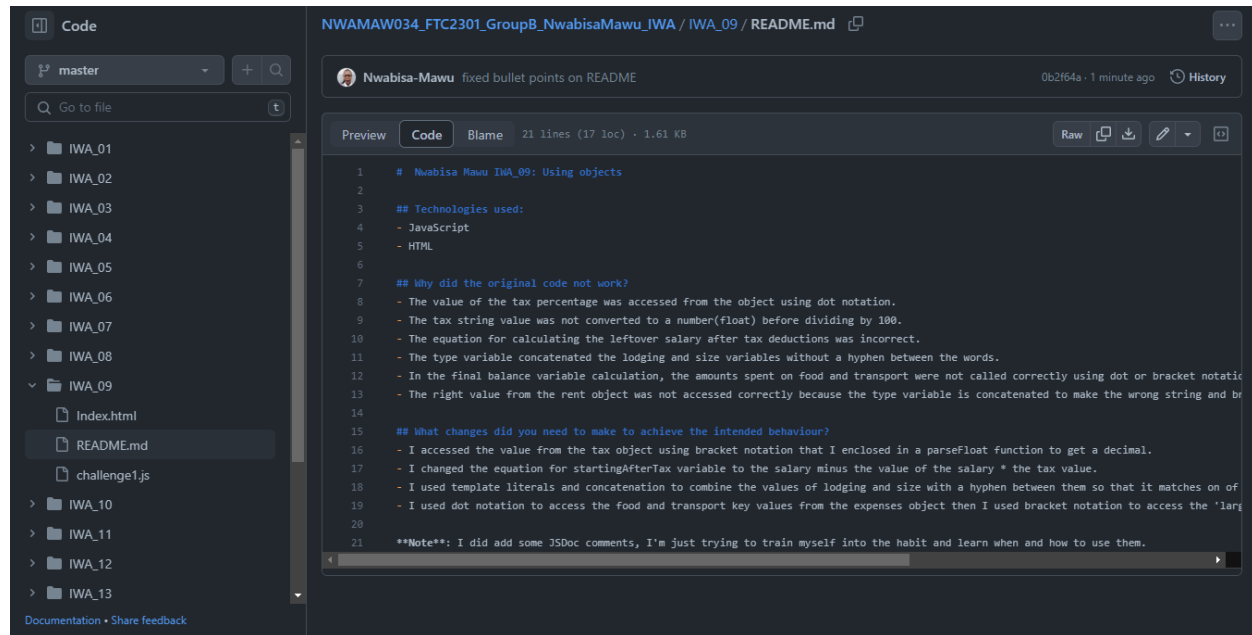


WA_03.4 Knowledge Check_DWA3.1

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 the content of README.md. The code is written in Markdown and includes a title, a section for technologies used, a section explaining why original code failed, and a section for 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 README.md file. The preview shows the title, the technologies used section, the why original code failed section, and the changes made to achieve the intended behavior section. The preview is as follows:

Nwabisa Mawu IWA_09: Using objects

Technologies used:

- JavaScript
- HTML

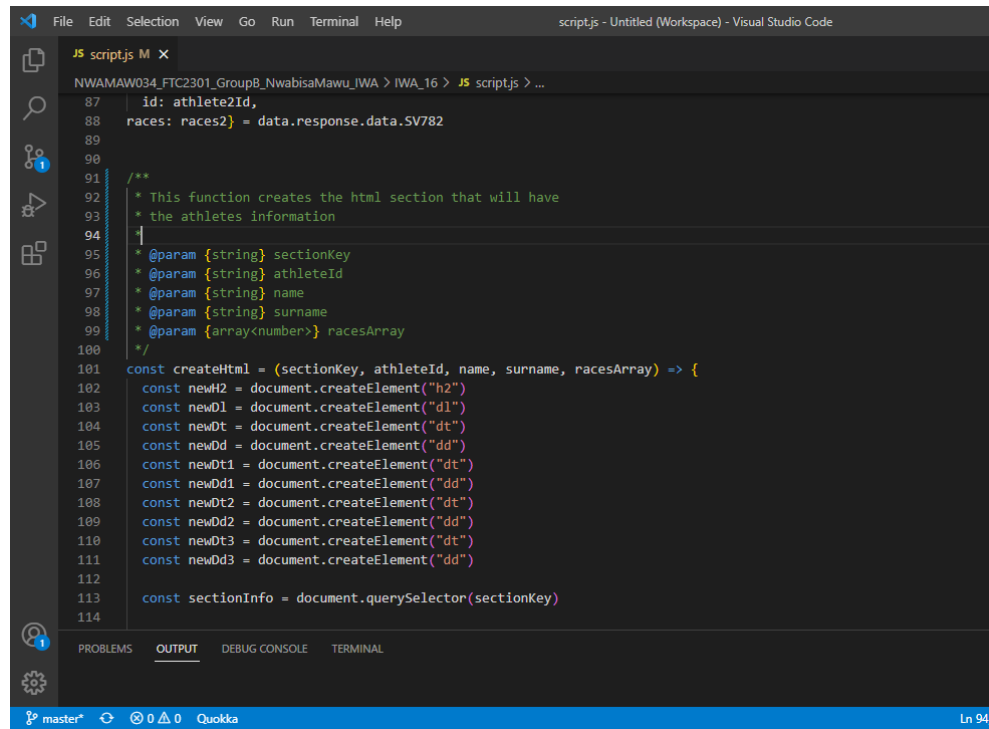
Why did the original code not work?

- 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.

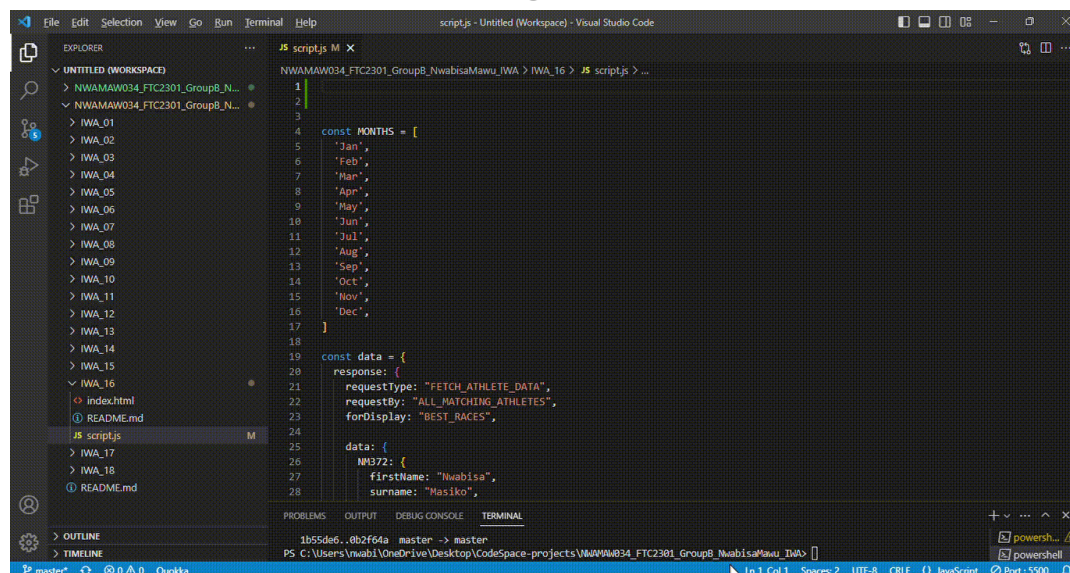
2. Please show how you applied JSDoc Comments to a piece of your code.



The screenshot shows a Visual Studio Code editor with a JavaScript file named `script.js`. The code defines a function `createHtml` that takes five parameters: `sectionKey`, `athleteId`, `name`, `surname`, and `racessArray`. The function is annotated with JSDoc comments. The first comment block, starting at line 91, describes the function's purpose: "This function creates the html section that will have the athletes information". The second comment block, starting at line 94, lists the parameters with their types: `@param {string} sectionKey`, `@param {string} athleteId`, `@param {string} name`, `@param {string} surname`, and `@param {array<number>} racessArray`. The function body starts at line 101 and includes several `document.createElement` calls to create HTML elements, followed by a `document.querySelector` call to find the section by key.

```
87 | id: athlete2Id,
88 | races: races2} = data.response.data.SV782
89 |
90 |
91 | /**
92 |  * This function creates the html section that will have
93 |  * the athletes information
94 |  */
95 | @param {string} sectionKey
96 | @param {string} athleteId
97 | @param {string} name
98 | @param {string} surname
99 | @param {array<number>} racessArray
100 | */
101 | const createHtml = (sectionKey, athleteId, name, surname, racessArray) => {
102 |   const newH2 = document.createElement("h2")
103 |   const newD1 = document.createElement("d1")
104 |   const newDt = document.createElement("dt")
105 |   const newDd = document.createElement("dd")
106 |   const newDt1 = document.createElement("dt")
107 |   const newDd1 = document.createElement("dd")
108 |   const newDt2 = document.createElement("dt")
109 |   const newDd2 = document.createElement("dd")
110 |   const newDt3 = document.createElement("dt")
111 |   const newDd3 = document.createElement("dd")
112 |
113 |   const sectionInfo = document.querySelector(sectionKey)
114 | }
```

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

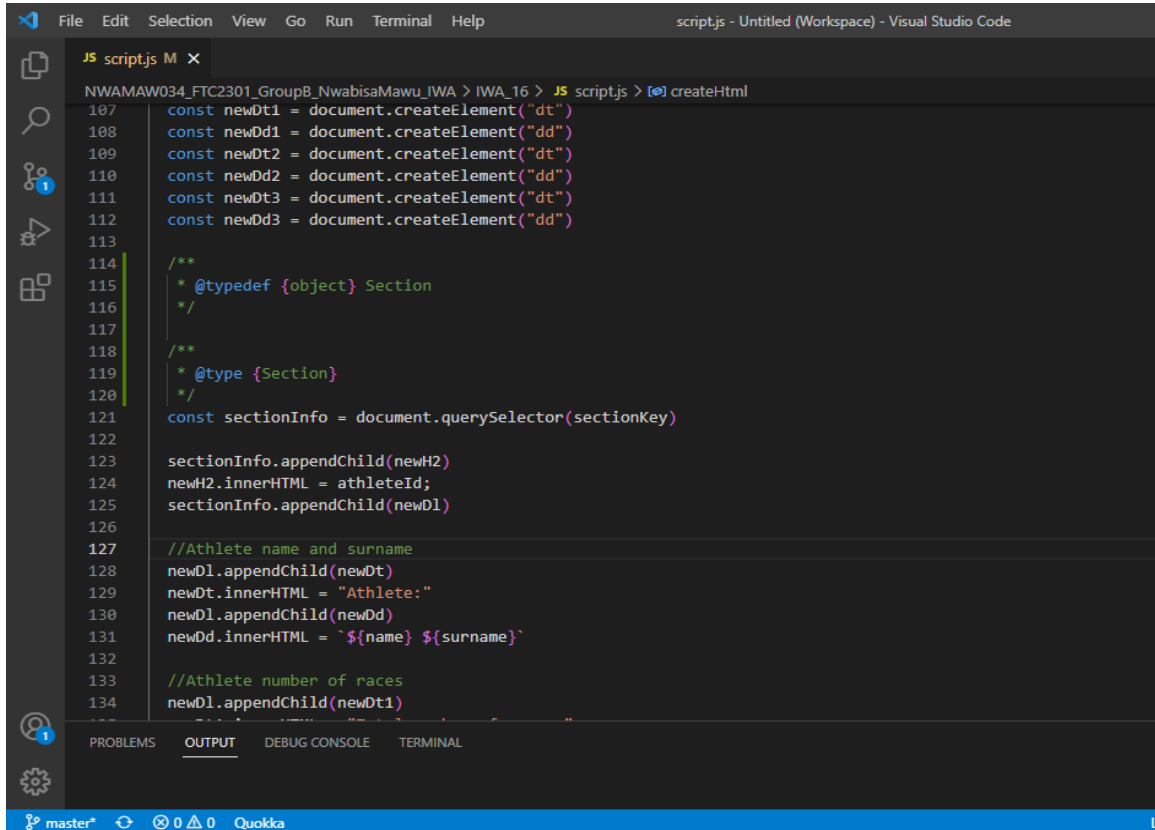


The screenshot shows a Visual Studio Code editor with a JavaScript file named `script.js`. The code defines a `MONTHS` array and a `data` object. The `data` object has a `response` property with `requestType`, `requestBy`, and `forDisplay` fields, and a `data` property with a `NW372` object containing `firstName` and `surname` fields. The file is annotated with the `@ts-check` annotation at the top. The Visual Studio Code interface shows the Explorer view on the left with the file structure, the Output view at the bottom, and the Terminal view at the bottom right.

```
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 |
20 | const data = {
21 |   response: {
22 |     requestType: "FETCH_ATHLETE_DATA",
23 |     requestBy: "ALL_MATCHING_ATHLETES",
24 |     forDisplay: "BEST_RACES",
25 |
26 |     data: {
27 |       NW372: {
28 |         firstName: "Nwabisa",
29 |         surname: "Masiko",
30 |       }
31 |     }
32 |   }
33 | }
```

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.



```
File Edit Selection View Go Run Terminal Help script.js - Untitled (Workspace) - Visual Studio Code

JS script.js M X
NWAMAW034_FTC2301_GroupB_NwabisaMawu_IWA > IWA_16 > JS script.js > createHtml
107 const newDt1 = document.createElement("dt")
108 const newDd1 = document.createElement("dd")
109 const newDt2 = document.createElement("dt")
110 const newDd2 = document.createElement("dd")
111 const newDt3 = document.createElement("dt")
112 const newDd3 = document.createElement("dd")
113
114 /**
115  * @typedef {object} Section
116  */
117
118 /**
119  * @type {Section}
120  */
121 const sectionInfo = document.querySelector(sectionKey)
122
123 sectionInfo.appendChild(newH2)
124 newH2.innerHTML = athleteId;
125 sectionInfo.appendChild(newDl)
126
127 //Athlete name and surname
128 newDl.appendChild(newDt)
129 newDt.innerHTML = "Athlete:"
130 newDl.appendChild(newDd)
131 newDd.innerHTML = `${name} ${surname}`
132
133 //Athlete number of races
134 newDl.appendChild(newDt1)
```