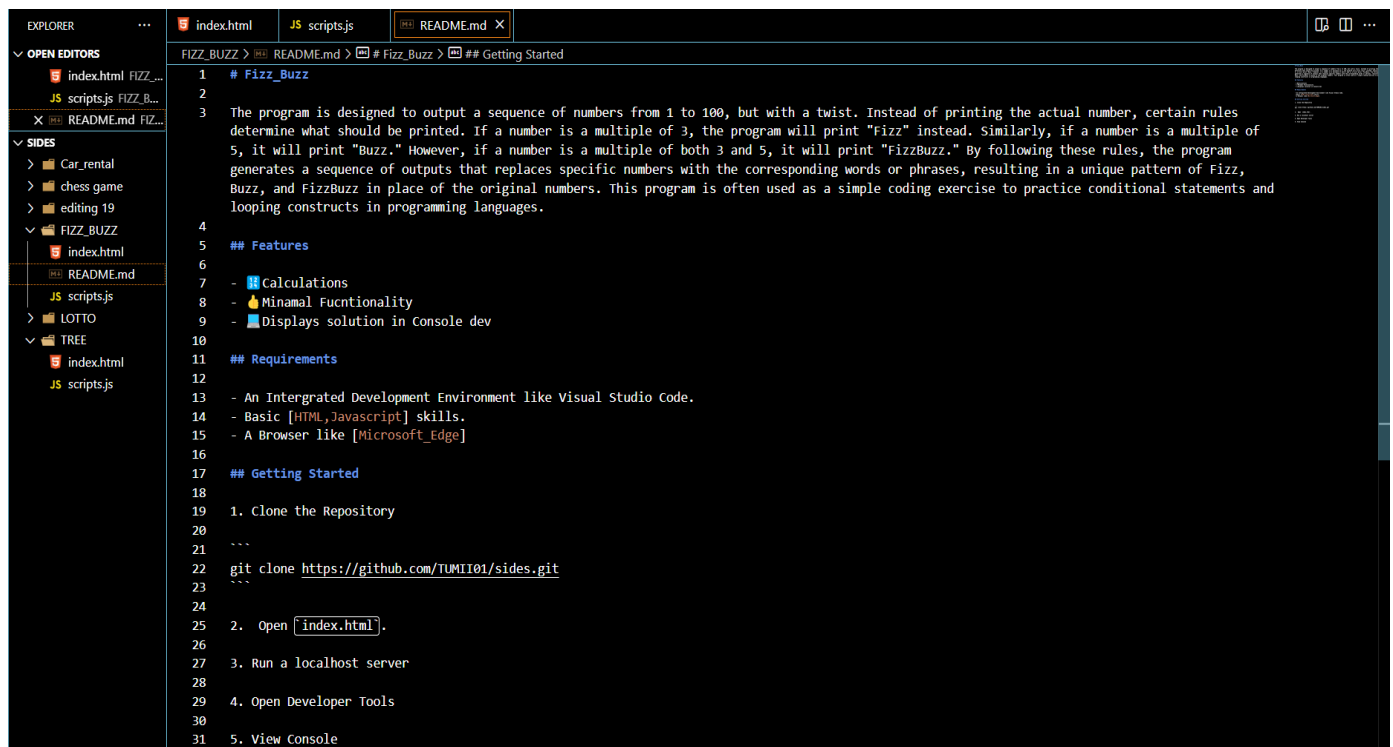


DWA_03.4 Knowledge Check_DWA3.1

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



The screenshot shows a code editor with a sidebar on the left and a main editor area on the right. The sidebar has two sections: 'OPEN EDITORS' and 'SIDES'. The 'OPEN EDITORS' section shows three files: 'index.html', 'JS scripts.js', and 'README.md'. The 'SIDES' section shows a tree view of a project structure with folders like 'Car_rental', 'chess game', 'editing 19', 'FIZZ_BUZZ', 'LOTTO', and 'TREE'. The 'FIZZ_BUZZ' folder is expanded, showing 'index.html' and 'README.md'. The 'README.md' file is selected and its content is displayed in the main editor area. The content of the 'README.md' file is as follows:

```
1 # Fizz_Buzz
2
3 The program is designed to output a sequence of numbers from 1 to 100, but with a twist. Instead of printing the actual number, certain rules
4 determine what should be printed. If a number is a multiple of 3, the program will print "fizz" instead. Similarly, if a number is a multiple of
5 5, it will print "Buzz." However, if a number is a multiple of both 3 and 5, it will print "FizzBuzz." By following these rules, the program
6 generates a sequence of outputs that replaces specific numbers with the corresponding words or phrases, resulting in a unique pattern of Fizz,
7 Buzz, and FizzBuzz in place of the original numbers. This program is often used as a simple coding exercise to practice conditional statements and
8 looping constructs in programming languages.
9
10 ## Features
11
12 - Calculations
13 - Minimal Funtionality
14 - Displays solution in Console dev
15
16 ## Requirements
17
18 - An Intergrated Development Environment like Visual Studio Code.
19 - Basic [HTML, Javascript] skills.
20 - A Browser like [Microsoft_Edge]
21
22 ## Getting Started
23
24 1. Clone the Repository
25
26 ```
27 git clone https://github.com/TUMI01/sides.git
28 ```
29
30 2. Open [index.html].
31
32 3. Run a localhost server
33
34 4. Open Developer Tools
35
36 5. View Console
```

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

```
3
4 // Set initial variables
5 const matches = books
6 const page = 1;
7
8
9 let startIndex = 0
10 let endIndex = 36
11 const extracted = books.slice(startIndex, endIndex)
12
13 // Get the menu element and create a fragment to append elements to
14 const menu = document.querySelector('[data-list-items]')
15 const fragment = document.createDocumentFragment()
16
17 // Loop through the extracted books and create an element for each
18 for (const { author, image, title, id, description, published } of extracted) {
19   let element = document.createElement('button')
20   element.classList = 'preview'
21   element.dataset.id = id
22   element.dataset.title = title
23   element.dataset.description = description
24   element.dataset.image = image
25   element.dataset.subtitle = (`${authors[author]} (${(new Date(published)).getFullYear()})`)
26   element.setAttribute('data-preview', id)
27   element.innerHTML = /* html */ `
28     <div></div>
32     <div class="preview__info">
33       |   <h3 class="preview__title">${title}</h3>
34       |   <div class="preview__author">${authors[author]}</div>
35     </div>`
36
37 // Append the fragment to the menu
38 | fragment.appendChild(element)
39 }
40 menu.appendChild(fragment)
41
```

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

```
92
93 //Dragging events
94 /**
95  * A handler that fires when a user drags over any element inside a column. In
96  * order to determine which column the user is dragging over the entire event
97  * bubble path is checked with `event.path` (or `event.composedPath()` for
98  * browsers that don't support `event.path`). The bubbling path is looped over
99  * until an element with a `data-area` attribute is found. Once found both the
100  * active dragging column is set in the `state` object in "data.js" and the HTML
101  * is updated to reflect the new column.
102  *
103  * @param {Event} event
104  */
105 const handleDragOver = (event) => {
106   event.preventDefault();
107   const path = event.path || event.composedPath();
108   let column = null;
109   for (const element of path) {
110     const { area } = element.dataset;
111     if (area) {
112       column = area;
113       break;
114     }
115   }
116   if (!column) return;
117   updateDragging({ over: column });
118   updateDraggingHtml({ over: column });
119 };
120 let dragged;
121 const handleDragStart = (e) => {
122   dragged = e.target;
123 };
124 const handleDragDrop = (f) => {
125   f.target.append(dragged);
126 };
```

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

```
1
2  export const BOOKS_PER_PAGE = 36;
3
4  > export const authors = { ...
92  }
93
94  > export const genres = { ...
163  }
164
165  > export const books = [ ...
25413 ]
```

```
// Import data and constants
import { BOOKS_PER_PAGE, authors, genres, books } from './data.js'

// Set initial variables
const matches = books
const page = 1;
```
