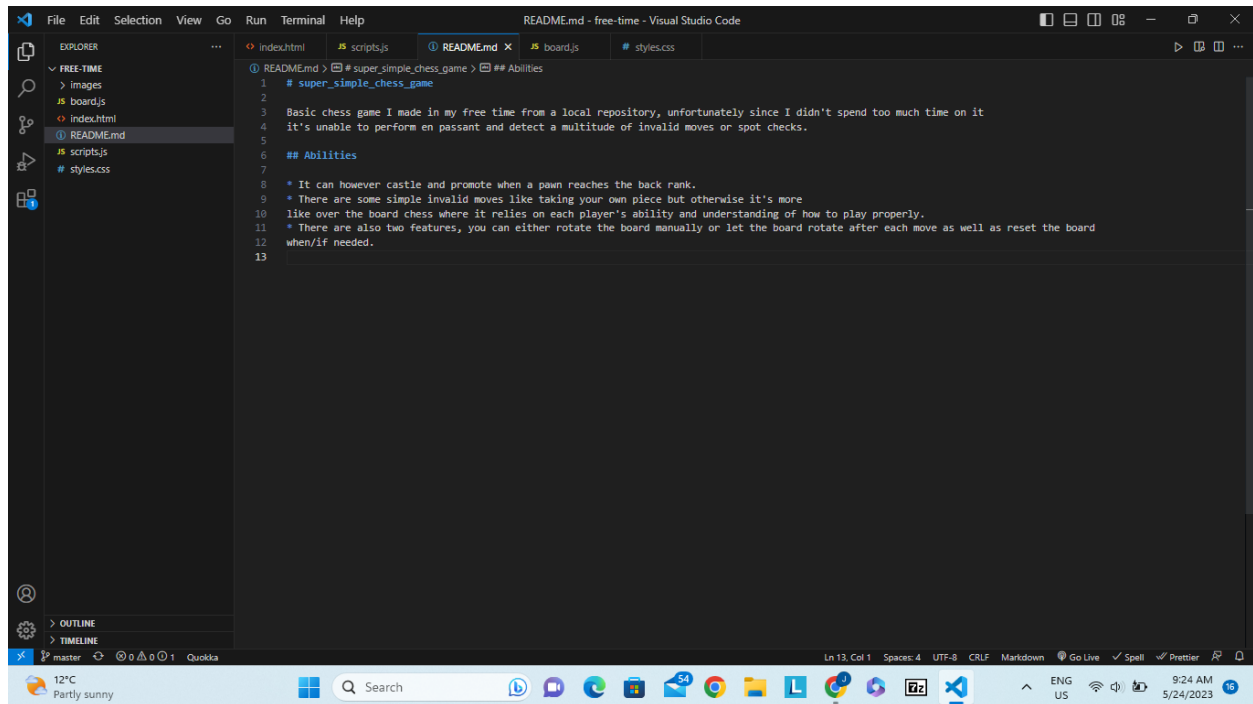


# DWA\_03.4 Knowledge Check\_DWA3.1

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



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

```
49 document.querySelector('.overlay__button').style.outline = 0; // Fixing the outline bug with the overlay cause outline
50
51
52
53 /**
54  * Function made to insert values from the inputted objects to the newly created 'option' element, a fragment is created
55  * as well and the new options elements with their values are appended to the fragment, the function then returns the
56  * fragment which can then be appended to the required parent element, the function also takes in a string argument
57  * which is primarily used to create a default option value i.e "All". To get the values from the objects the function
58  * runs a loop through the object getting both the key values and the properties of those key values.
59  * @param {string} text
60  * @param {object} object
61  * @returns appended element
62  */
63 const optionsCreate = (text, object) => {
64   const fragment = document.createDocumentFragment();
65   const allOption = document.createElement("option");
66   allOption.value = "All";
67   allOption.innerText = text;
68   fragment.appendChild(allOption);
69
70   for (const [keyValue, property] of Object.entries(object)) {
71     const option = document.createElement("option");
72     option.value = keyValue;
73     option.innerText = property;
74     fragment.appendChild(option);
75   }
76   return fragment;
77 };
78
79 selectors.genresSelect.appendChild(optionsCreate("All genres", genresObj));
80 selectors.authorSelect.appendChild(optionsCreate("All authors", authors));
81
82 //Set the colors of the preview overlay text to correspond with the theme change
83 selectors.previewOverlay.titleOverlay.style.color = `rgba(var(--color-dark))`
84 selectors.previewOverlay.dataOverlay.style.color = `rgba(var(--color-dark))`
85 selectors.previewOverlay.infoOverlay.style.color = `rgba(var(--color-dark))`
```

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

```
1 import { genresObj, authors } from './data.js';
2 // @ts-check
3
4
5 /**
6  * Object containing all query selectors
7  */
8 export const selectors = {
9   list: document.querySelector("[data-list-items]"),
10   message: document.querySelector("[data-list-message]"),
11   loadMore: document.querySelector("[data-list-button]"),
12   previewOverlay: {
13     overlay: document.querySelector("[data-list-active]"),
14     overlayBtn: document.querySelector("[data-list-close]"),
15     overlayBlun: document.querySelector("[data-list-blun]"),
16     overlayImage: document.querySelector("[data-list-image]"),
17     titleOverlay: document.querySelector("[data-list-title]"),
18     dataOverlay: document.querySelector("[data-list-subtitle]"),
19     infoOverlay: document.querySelector("[data-list-description]"),
20   },
21   theme: {
22     themeBtn: document.querySelector("[data-header-settings]"),
23     themeOverlay: document.querySelector("[data-settings-overlay]"),
24     themeCancelBtn: document.querySelector("[data-settings-cancel]"),
25     themeForm: document.querySelector("[data-settings-form]"),
26     themeSelect: document.querySelector("[data-settings-theme]"),
27   },
28   search: {
29     searchBtn: document.querySelector("[data-header-search]"),
30     searchOverlay: document.querySelector("[data-search-overlay]"),
31     searchCancelBtn: document.querySelector("[data-search-cancel]"),
32     searchForm: document.querySelector("[data-search-form]"),
33   },
34   genresSelect: document.querySelector("[data-search-genres]"),
35   authorSelect: document.querySelector("[data-search-authors]"),
36   title: document.querySelector("[data-search-title]"),
37 };
38
39 export const css = {
```

---

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

---