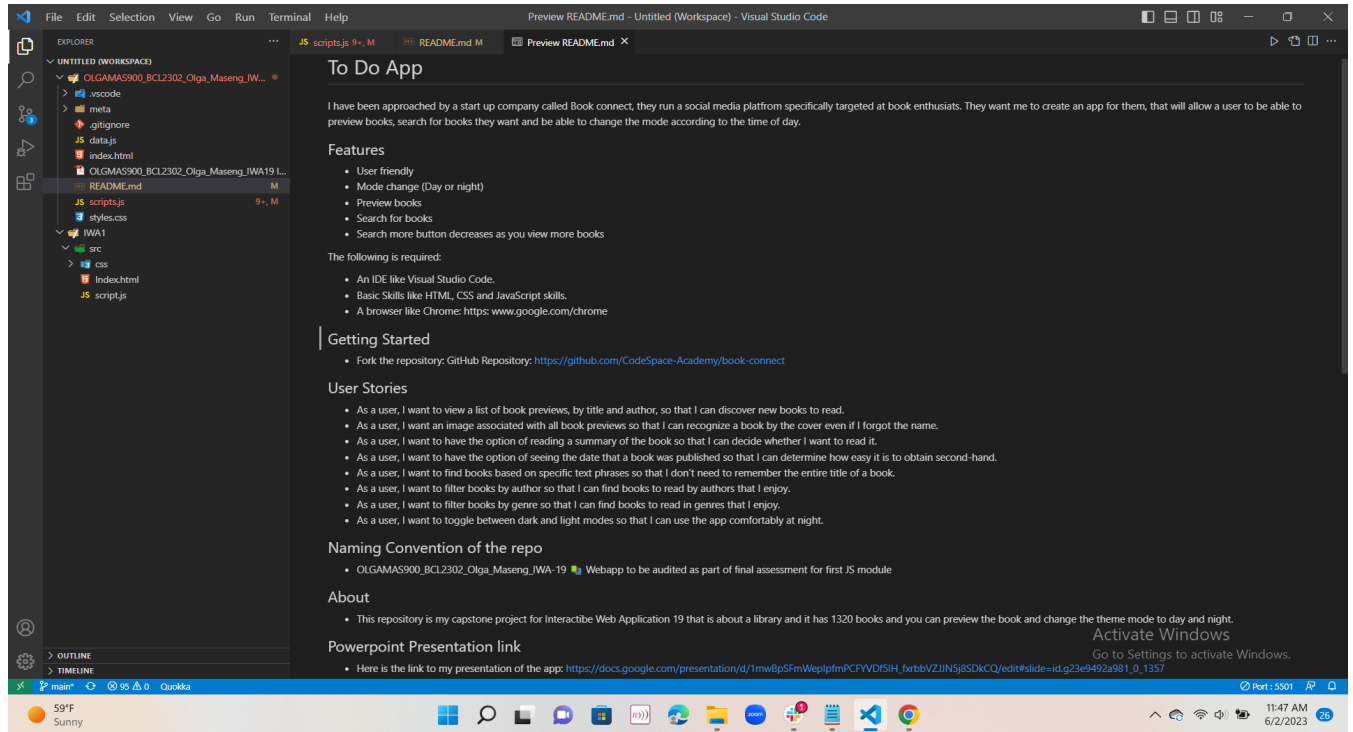
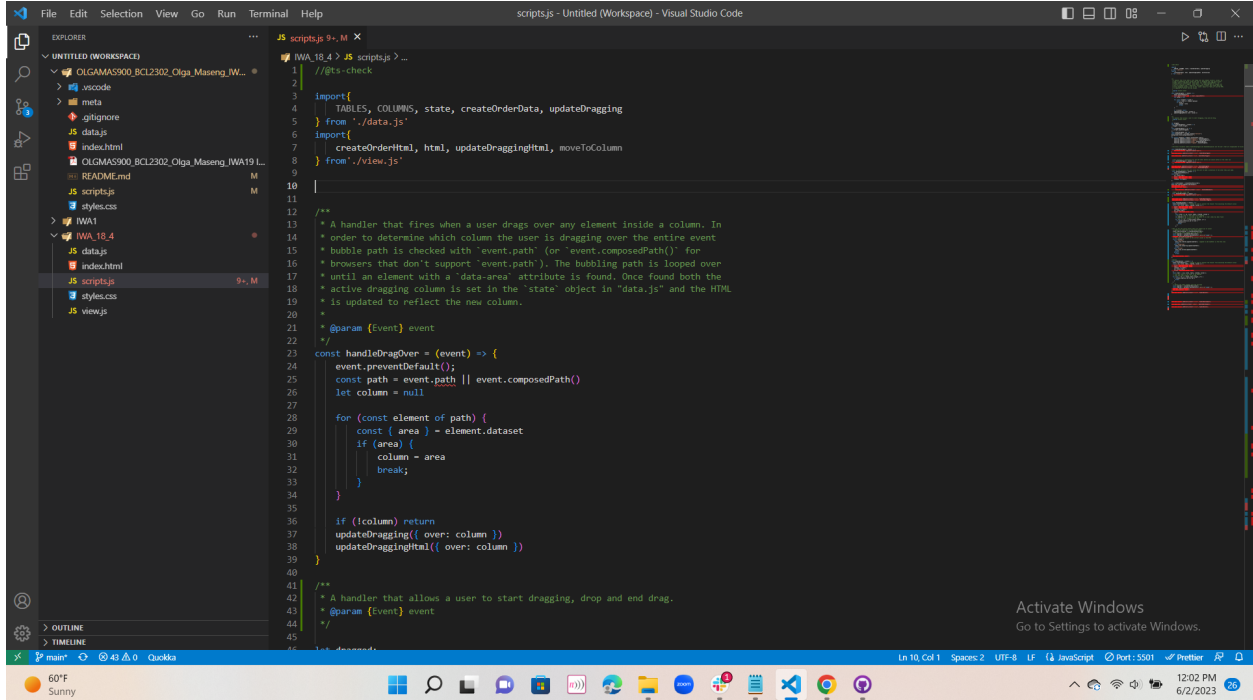


DWA_03.4 Knowledge Check_DWA3.1

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.

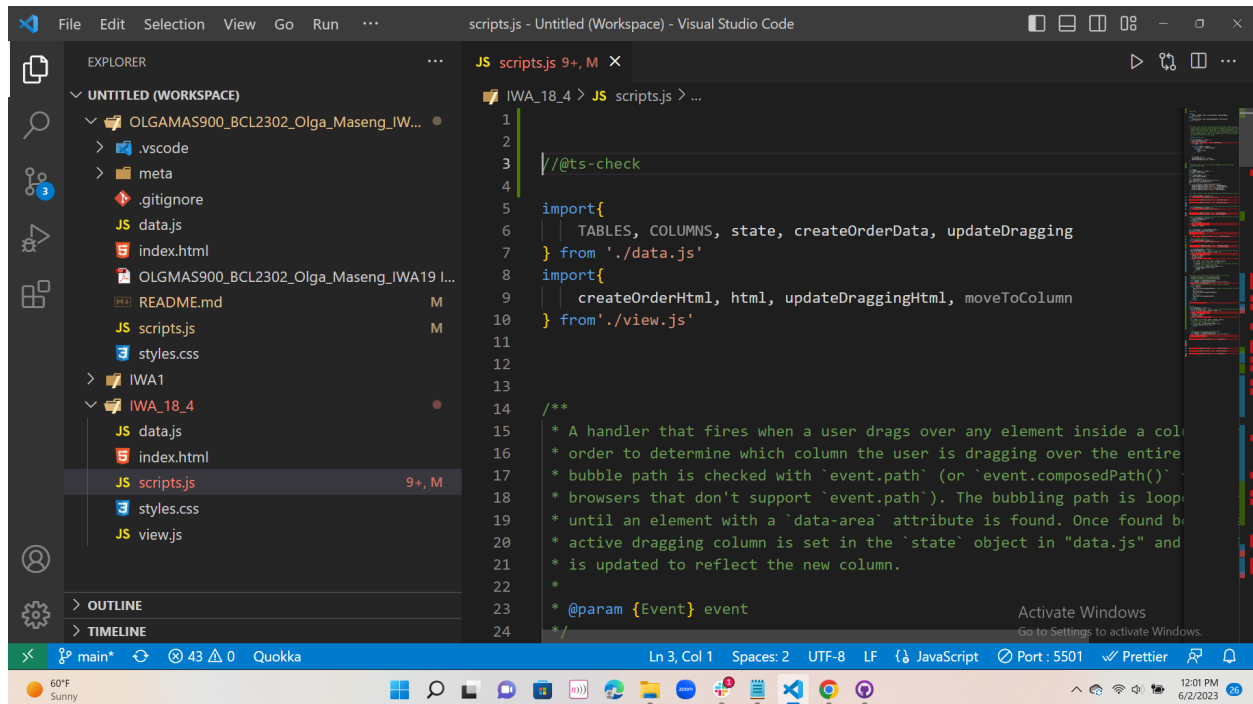


The screenshot shows the Visual Studio Code editor with a workspace named 'scripts.js - Untitled (Workspace)'. The Explorer panel on the left shows a project structure with folders like 'OLGAMAS900_BCL2302_Olga_Maseng_IWA19 L...' and 'IWA18.4'. The file 'scripts.js' is open in the editor. The code includes JSDoc comments for a function 'handleDragOver' and a parameter 'event'.

```
1 //Rx-check
2
3 import {
4   TABLES, COLUMNS, state, createOrderData, updateDragging
5 } from './data.js'
6 import {
7   createOrderHtml, html, updateDraggingHtml, moveToColumn
8 } from './view.js'
9
10
11
12
13 /**
14  * A handler that fires when a user drags over any element inside a column. In
15  * order to determine which column the user is dragging over the entire event
16  * bubble path is checked with 'event.path' (or 'event.composedPath()' for
17  * browsers that don't support 'event.path'). The bubbling path is looped over
18  * until an element with a 'data-area' attribute is found. Once found both the
19  * active dragging column is set in the 'state' object in 'data.js' and the HTML
20  * is updated to reflect the new column.
21  *
22  * @param {Event} event
23  */
24 const handleDragOver = (event) => {
25   event.preventDefault();
26   const path = event.path || event.composedPath()
27   let column = null
28
29   for (const element of path) {
30     const { area } = element.dataset
31     if (area) {
32       column = area
33       break;
34     }
35   }
36
37   if (!column) return
38   updateDragging({ over: column })
39   updateDraggingHtml({ over: column })
40 }
41
42 /**
43  * A handler that allows a user to start dragging, drop and end drag.
44  *
45  * @param {Event} event
46  */
```

The status bar at the bottom shows 'Ln 10, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'JavaScript', 'Port: 5501', 'Prettier', and the system clock '12:02 PM 6/2/2023'.

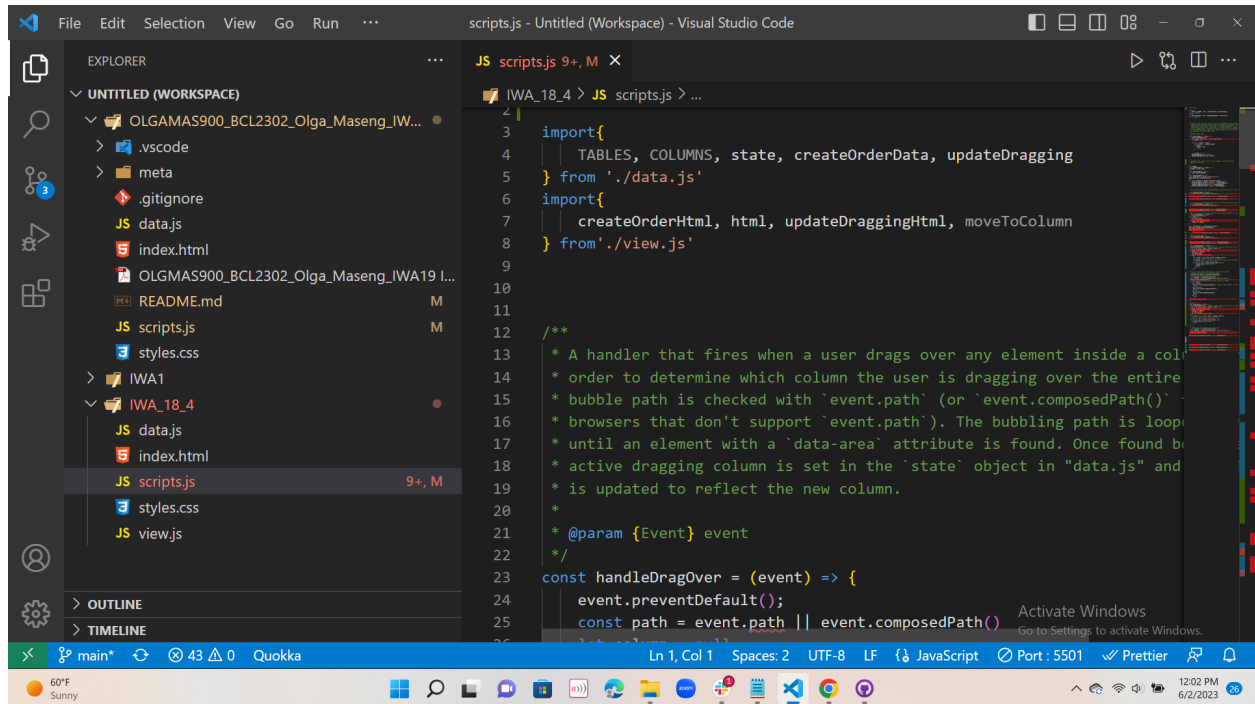
3. Please show how you applied the @ts-check annotation to a piece of your code.



The screenshot shows the Visual Studio Code interface with a workspace named 'scripts.js - Untitled (Workspace)'. The Explorer panel on the left shows a project structure with folders 'OLGAMAS900_BCL2302_Olga_Maseng_IW...' and 'IWA1', and subfolders 'IWA18_4' and 'IWA19'. The file 'scripts.js' is selected in the 'IWA18_4' folder. The main editor shows the content of 'scripts.js', which includes imports from './data.js' and './view.js', and a JSDoc comment for a function. The '@ts-check' annotation is applied to the file, as indicated by the 'IWA18_4 > JS scripts.js' tab and the 'IWA18_4 > JS scripts.js' file name in the editor. The status bar at the bottom shows 'Ln 3, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'JavaScript', 'Port: 5501', and 'Prettier'.

```
1
2
3 // @ts-check
4
5 import {
6   TABLES, COLUMNS, state, createOrderData, updateDragging
7 } from './data.js'
8 import {
9   createOrderHtml, html, updateDraggingHtml, moveToColumn
10 } from './view.js'
11
12
13
14 /**
15  * A handler that fires when a user drags over any element inside a col
16  * order to determine which column the user is dragging over the entire
17  * bubble path is checked with `event.path` (or `event.composedPath()`
18  * browsers that don't support `event.path`). The bubbling path is loop
19  * until an element with a `data-area` attribute is found. Once found b
20  * active dragging column is set in the `state` object in "data.js" and
21  * is updated to reflect the new column.
22  *
23  * @param {Event} event
24  */
```

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



```
1 2
3  import{
4      TABLES, COLUMNS, state, createOrderData, updateDragging
5  } from './data.js'
6  import{
7      createOrderHtml, html, updateDraggingHtml, moveToColumn
8  } from './view.js'
9
10
11
12  /**
13   * A handler that fires when a user drags over any element inside a column
14   * order to determine which column the user is dragging over the entire
15   * bubble path is checked with 'event.path' (or 'event.composedPath()'
16   * browsers that don't support 'event.path'). The bubbling path is looped
17   * until an element with a 'data-area' attribute is found. Once found by
18   * active dragging column is set in the 'state' object in "data.js" and
19   * is updated to reflect the new column.
20   */
21   @param {Event} event
22   */
23   const handleDragOver = (event) => {
24       event.preventDefault();
25       const path = event.path || event.composedPath();
26       let column = null;
```