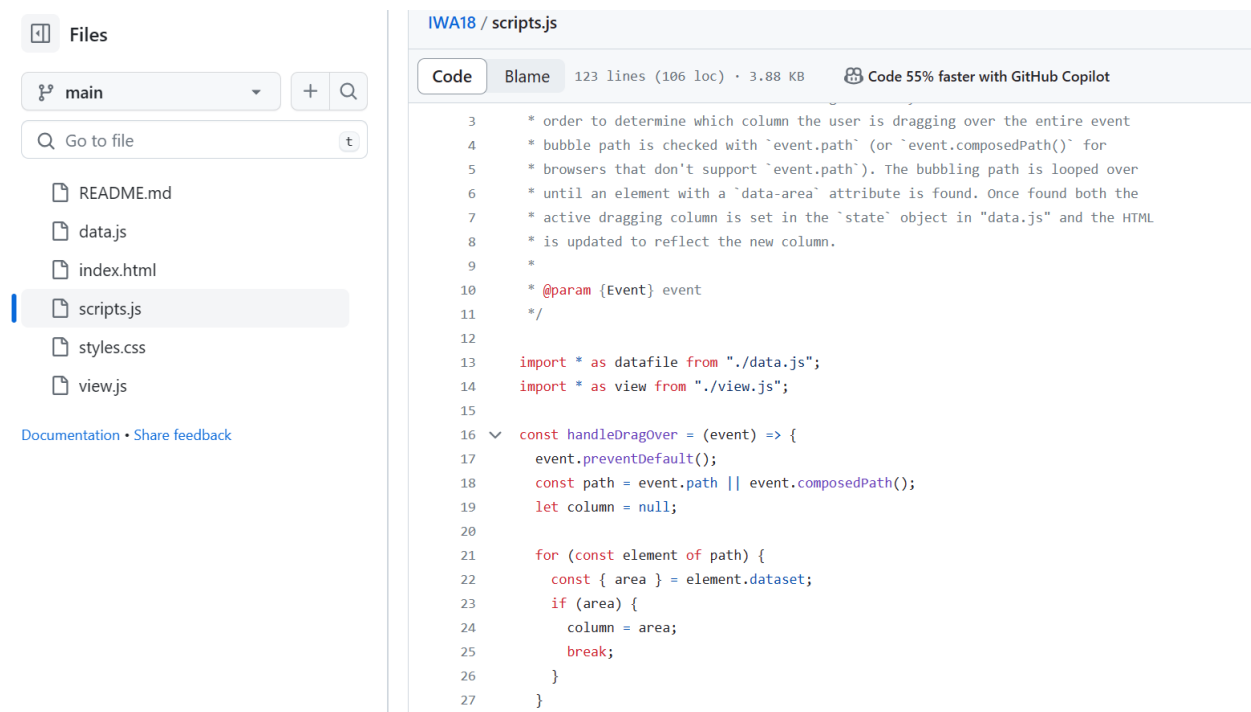


DWA_03.4 Knowledge Check_DWA3.1

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 displays a code editor interface. On the left, a file explorer shows a project structure with files: README.md, data.js, index.html, scripts.js (selected), styles.css, and view.js. Below the file list are links for "Documentation" and "Share feedback". The main editor area shows the content of the selected file, `scripts.js`. The code includes JSDoc comments and JavaScript code for handling drag events. The JSDoc comments describe the purpose of the code, such as determining which column the user is dragging over and checking the bubble path. The JavaScript code defines a function `handleDragOver` that takes an event object and returns a boolean. It uses `event.preventDefault()` to prevent the default drag behavior and checks the `event.path` property to determine the column being dragged over. The code also imports `datafile` from `./data.js` and `view` from `./view.js`.

```
IWA18 / scripts.js
Code Blame 123 lines (106 loc) · 3.88 KB Code 55% faster with GitHub Copilot
3  * order to determine which column the user is dragging over the entire event
4  * bubble path is checked with `event.path` (or `event.composedPath()` for
5  * browsers that don't support `event.path`). The bubbling path is looped over
6  * until an element with a `data-area` attribute is found. Once found both the
7  * active dragging column is set in the `state` object in "data.js" and the HTML
8  * is updated to reflect the new column.
9  *
10 * @param {Event} event
11 */
12
13 import * as datafile from "./data.js";
14 import * as view from "./view.js";
15
16 const handleDragOver = (event) => {
17   event.preventDefault();
18   const path = event.path || event.composedPath();
19   let column = null;
20
21   for (const element of path) {
22     const { area } = element.dataset;
23     if (area) {
24       column = area;
25       break;
26     }
27   }
28 }
```

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

```

@ts-check
/**
 * A handler that fires when a user drags over any element inside a column. In
 * order to determine which column the user is dragging over the entire event
 * bubble path is checked with `event.path` (or `event.composedPath()` for
 * browsers that don't support `event.path`). The bubbling path is looped over
 * until an element with a `data-area` attribute is found. Once found both the
 * active dragging column is set in the `state` object in "data.js" and the HTML
 * is updated to reflect the new column.
 *
 * @param {Event} event
 */
import * as datafile from "./data.js";    "datafile": Unknown word.
import * as view from "./view.js";

```

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

```

12  /**
13  * @typedef {Object} Order
14  * @property {String} title - The title of the order
15  * @property {String} table - The associated table for the order
16  * @property {String} column - The column where the order is placed
17  * @property {String} id - The unique identifier for the order
18  */

```
