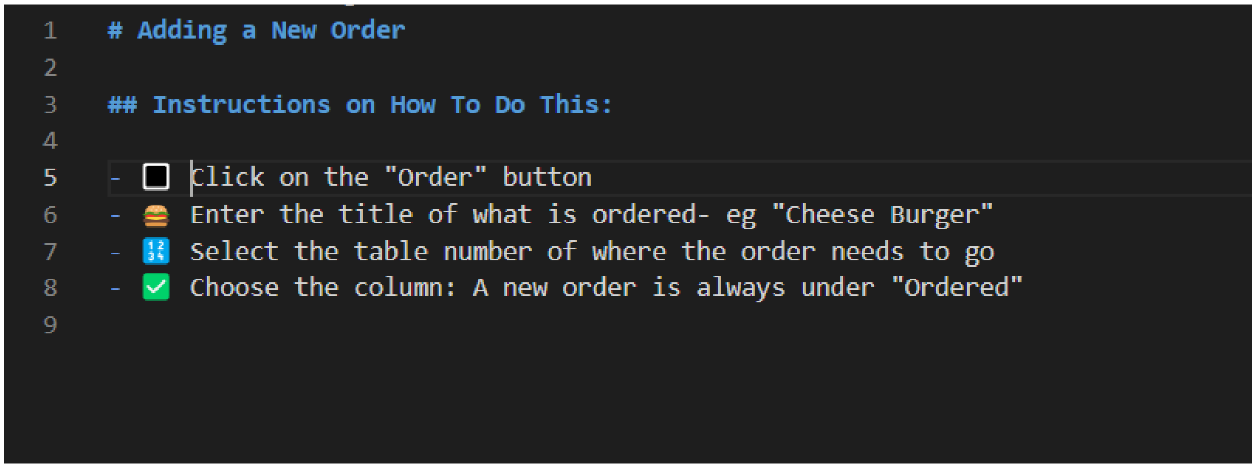
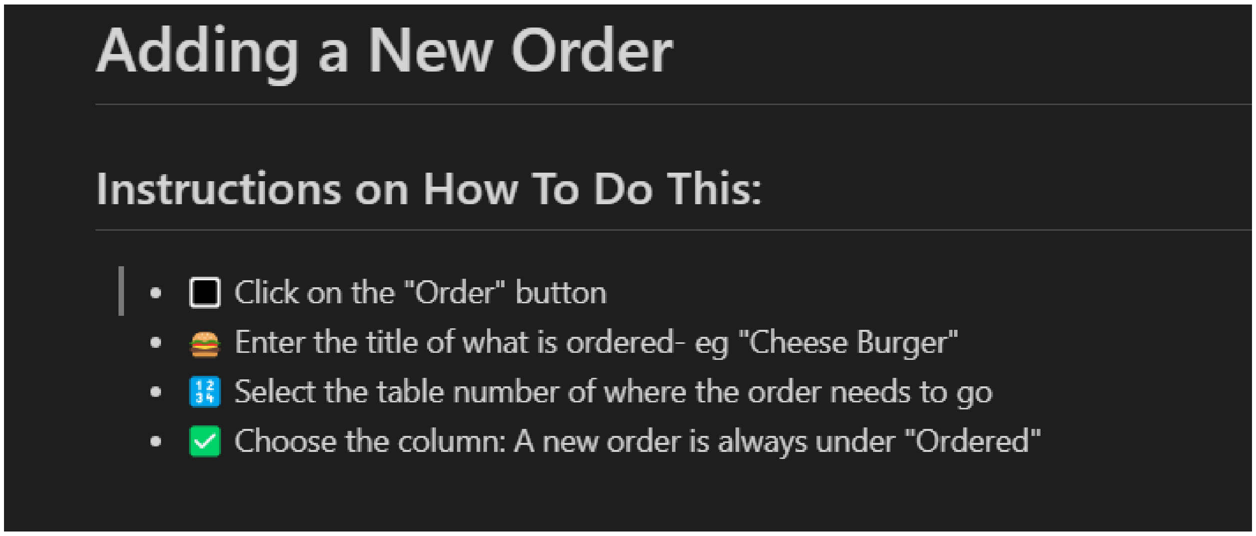
DWA_03.4 Knowledge Check_DWA3.1

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



```
# Adding a New Order

## Instructions on How To Do This:

- ⬛ Click on the "Order" button
- 🍔 Enter the title of what is ordered- eg "Cheese Burger"
- 🔢 Select the table number of where the order needs to go
- ✅ Choose the column: A new order is always under "Ordered"

```

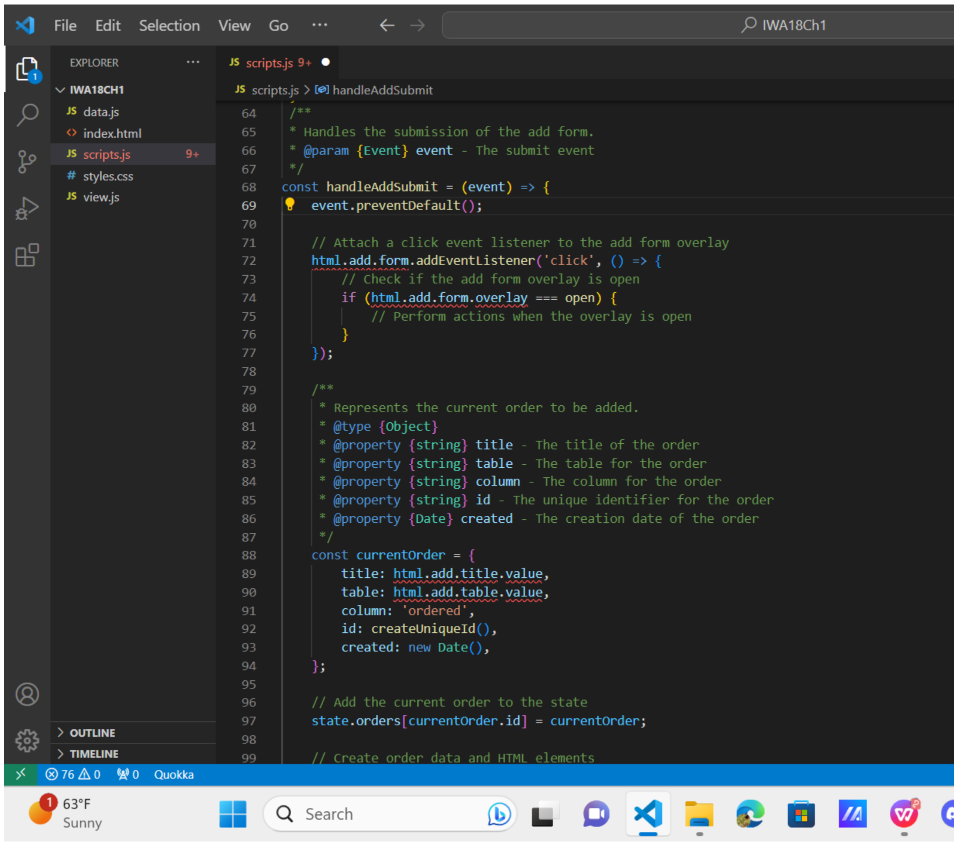


Adding a New Order

Instructions on How To Do This:

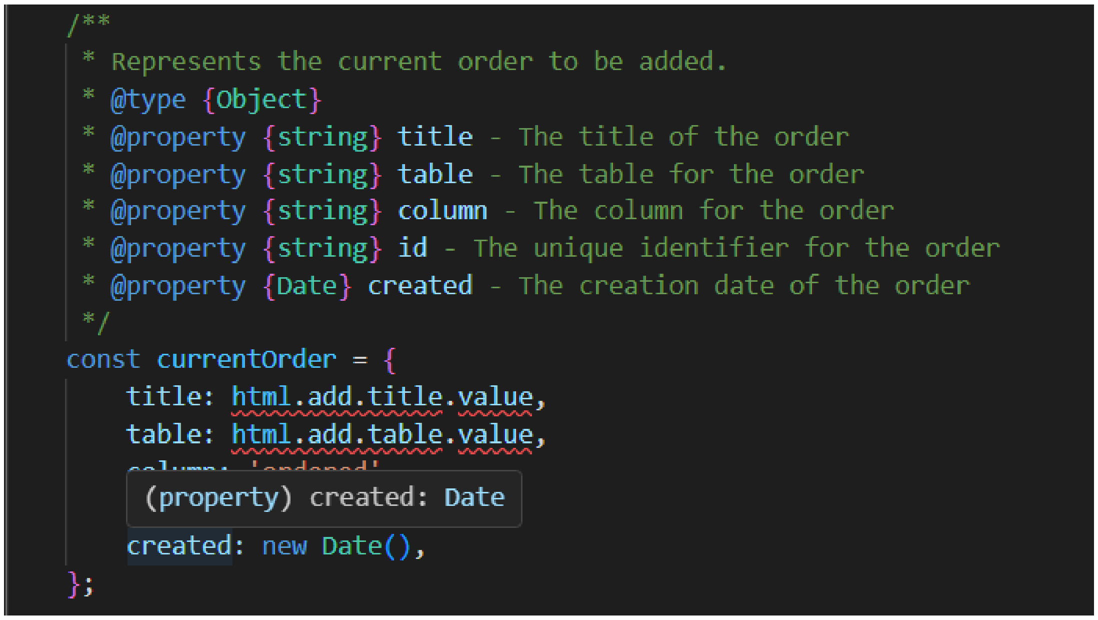
- ⬛ Click on the "Order" button
- 🍔 Enter the title of what is ordered- eg "Cheese Burger"
- 🔢 Select the table number of where the order needs to go
- ✅ Choose the column: A new order is always under "Ordered"

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



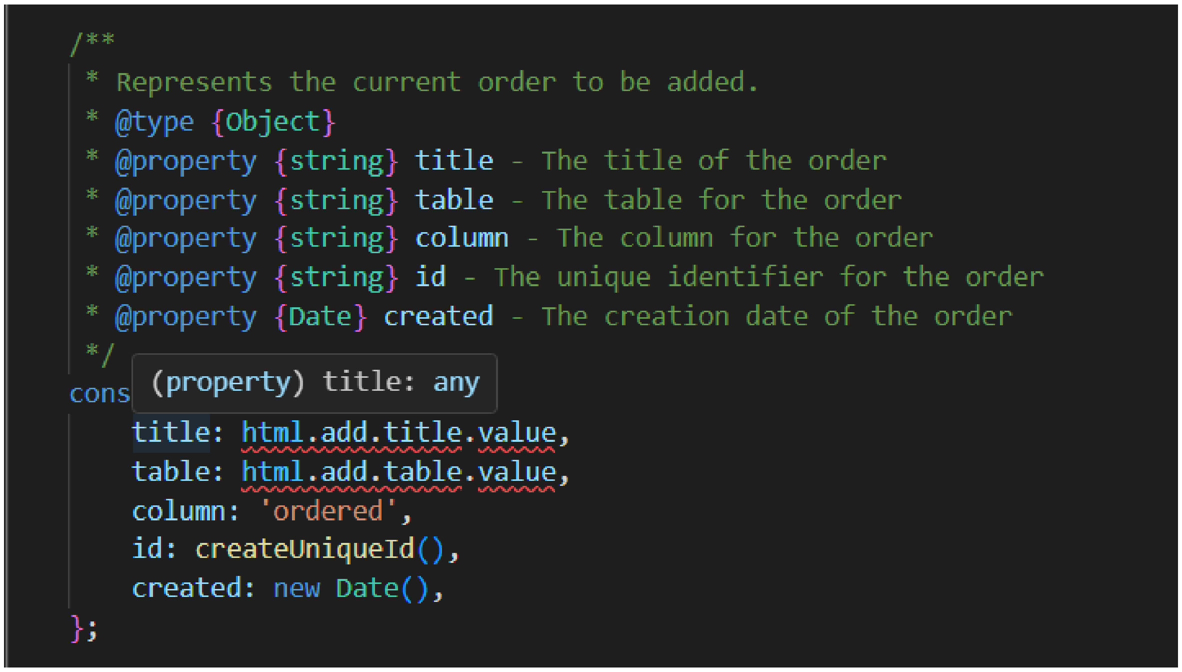
```javascript
/**
* Handles the submission of the add form.
* @param {Event} event - The submit event
*/
const handleAddSubmit = (event) => {
    event.preventDefault();

    // Attach a click event listener to the add form overlay
    html.add.form.addEventListener('click', () => {
        // Check if the add form overlay is open
        if (html.add.form.overlay === open) {
            // Perform actions when the overlay is open
        }
    });

    /**
     * Represents the current order to be added.
     * @type {Object}
     * @property {string} title - The title of the order
     * @property {string} table - The table for the order
     * @property {string} column - The column for the order
     * @property {string} id - The unique identifier for the order
     * @property {Date} created - The creation date of the order
     */
    const currentOrder = {
        title: html.add.title.value,
        table: html.add.table.value,
        column: 'ordered',
        id: createUniqueId(),
        created: new Date(),
    };

    // Add the current order to the state
    state.orders[currentOrder.id] = currentOrder;

    // Create order data and HTML elements
```

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



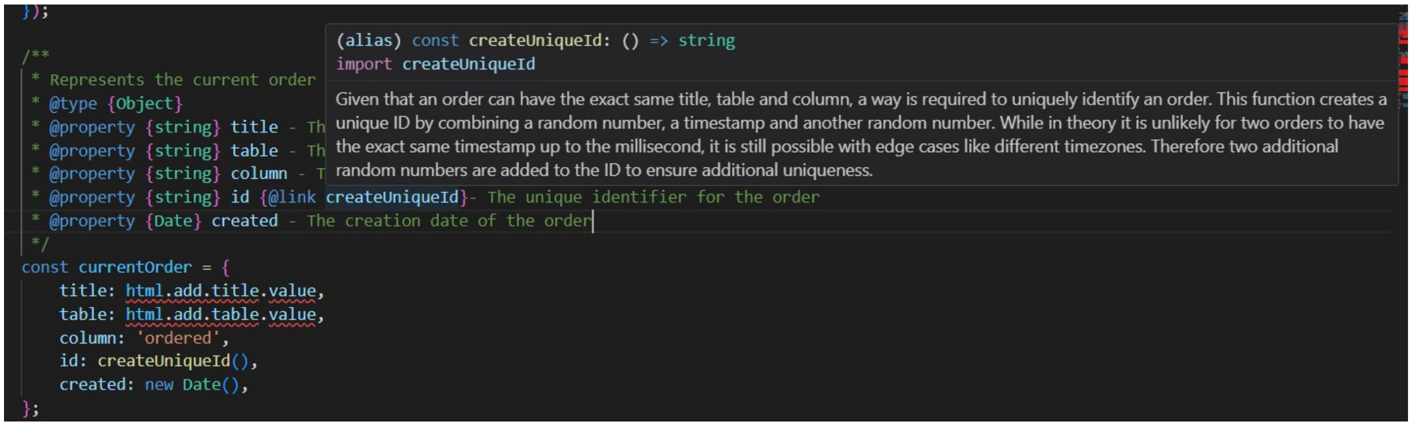
```
/**
 * Represents the current order to be added.
 * @type {Object}
 * @property {string} title - The title of the order
 * @property {string} table - The table for the order
 * @property {string} column - The column for the order
 * @property {string} id - The unique identifier for the order
 * @property {Date} created - The creation date of the order
 */
const currentOrder = {
    title: html.add.title.value,
    table: html.add.table.value,
    column: 'ordered'
    (property) created: Date
    created: new Date(),
};
```

Here the created line is not underlined as an error because it is actually a date which is what it needs to be, but below the title is underlined as an error because the way this is written the right hand side could be anything and not only a string, even though the value of that side of the argument is in fact a string.



```
/**
 * Represents the current order to be added.
 * @type {Object}
 * @property {string} title - The title of the order
 * @property {string} table - The table for the order
 * @property {string} column - The column for the order
 * @property {string} id - The unique identifier for the order
 * @property {Date} created - The creation date of the order
 */
cons (property) title: any
    title: html.add.title.value,
    table: html.add.table.value,
    column: 'ordered',
    id: createUniqueId(),
    created: new Date(),
};
```

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



```
});

/**
 * Represents the current order
 * @type {Object}
 * @property {string} title - Th
 * @property {string} table - Th
 * @property {string} column - T
 * @property {string} id {@link createUniqueId}- The unique identifier for the order
 * @property {Date} created - The creation date of the order
 */
const currentOrder = {
    title: html.add.title.value,
    table: html.add.table.value,
    column: 'ordered',
    id: createUniqueId(),
    created: new Date(),
};
```

```
(alias) const createUniqueId: () => string
import createUniqueId
```

Given that an order can have the exact same title, table and column, a way is required to uniquely identify an order. This function creates a unique ID by combining a random number, a timestamp and another random number. While in theory it is unlikely for two orders to have the exact same timestamp up to the millisecond, it is still possible with edge cases like different timezones. Therefore two additional random numbers are added to the ID to ensure additional uniqueness.

I added a link to a function for creating the unique order ID.