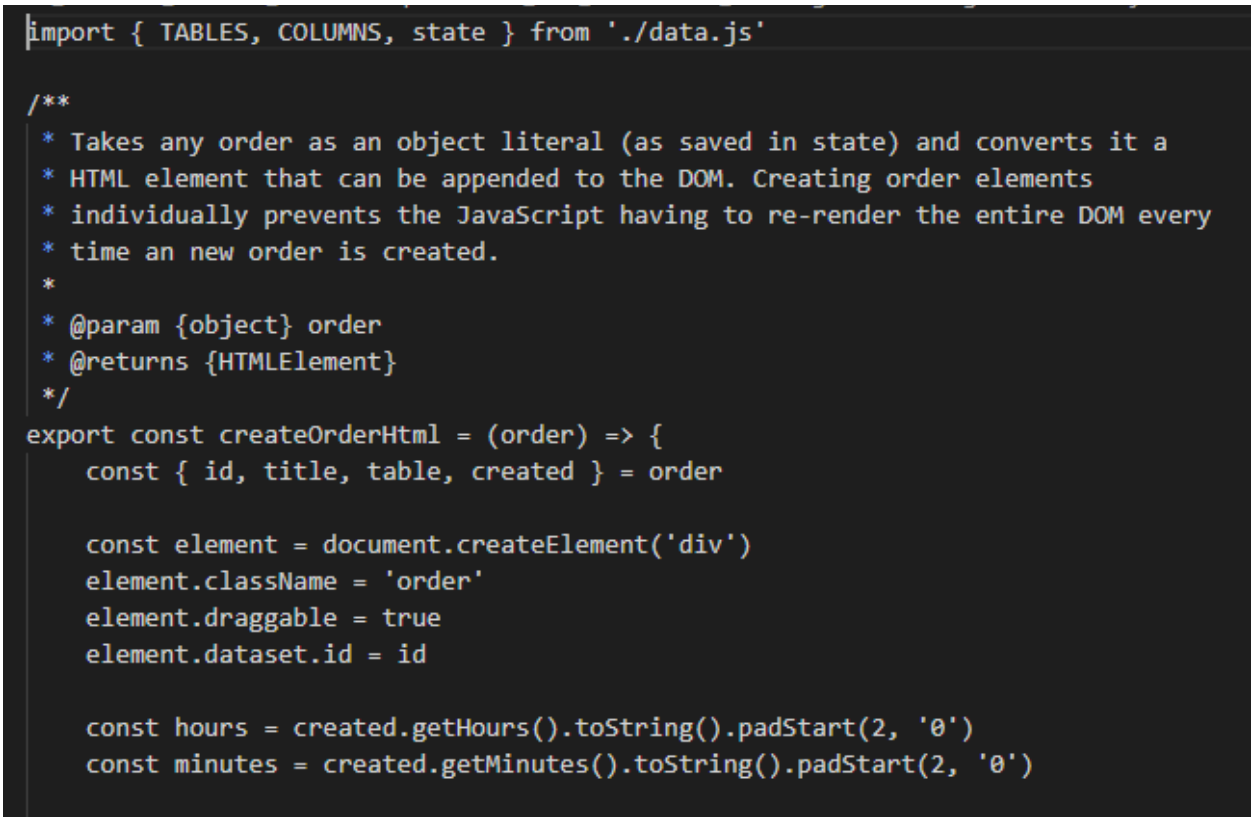
DWA_03.4 Knowledge Check_DWA3.1

---

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



```
import { TABLES, COLUMNS, state } from './data.js'

/**
 * Takes any order as an object literal (as saved in state) and converts it a
 * HTML element that can be appended to the DOM. Creating order elements
 * individually prevents the JavaScript having to re-render the entire DOM every
 * time an new order is created.
 *
 * @param {object} order
 * @returns {HTMLElement}
 */
export const createOrderHtml = (order) => {
    const { id, title, table, created } = order

    const element = document.createElement('div')
    element.className = 'order'
    element.draggable = true
    element.dataset.id = id

    const hours = created.getHours().toString().padStart(2, '0')
    const minutes = created.getMinutes().toString().padStart(2, '0')
```

---

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

```
import { TABLES, COLUMNS, state } from './data.js'

/**
 * Takes any order as an object literal (as saved in state) and converts it a
 * HTML element that can be appended to the DOM. Creating order elements
 * individually prevents the JavaScript having to re-render the entire DOM every
 * time an new order is created.
 *
 * @param {object} order
 * @returns {HTMLElement}
 */
export const createOrderHtml = (order) => {
    const { id, title, table, created } = order

    const element = document.createElement('div')
    element.className = 'order'
    element.draggable = true
    element.dataset.id = id

    const hours = created.getHours().toString().padStart(2, '0')
    const minutes = created.getMinutes().toString().padStart(2, '0')

    element.innerHTML = /* html */ `
        <div class="order__title" data-order-title>${title}</div>
```

---

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

```
import { TABLES, COLUMNS, state } from './data.js'

/**
 * Takes any order as an object literal (as saved in state) and converts it a
 * HTML element that can be appended to the DOM. Creating order elements
 * individually prevents the JavaScript having to re-render the entire DOM every
 * time an new order is created.
 *
 * @param {object} order
 * @returns {HTMLElement}
 */
export const createOrderHtml = (order) => {
    const { id, title, table, created } = order

    const element = document.createElement('div')
    element.className = 'order'
    element.draggable = true
    element.dataset.id = id

    const hours = created.getHours().toString().padStart(2, '0')
    const minutes = created.getMinutes().toString().padStart(2, '0')
```

---

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

TS Check Annotation :

```
/**
 * Takes a specific order HTML and clones it into memory. The original HTML
 * element is then removed from the DOM, while the cloned duplicate is added to
 * the bottom of the column that is specified.
 *
 * @param {string} id - The "id" value of a specific order object. Note that
 * only the "id" value is used, not the entire object.
 *
 * @param {string} newColumn - The name of the column that the order should be
 * moved to. This should coincide with one of the values present in the COLUMNS
 * array in "data.js"
 */
export const moveToColumn = (id, newColumn) => {
    const htmlSource = document.querySelector(`[data-id="${id}"]`)
    const duplicate = htmlSource.cloneNode(true)
    html.columns[newColumn].appendChild(duplicate)
    htmlSource.remove()
}

/**
 * Starts the app focused on the "add order" button. This means that users can
 * immediately started adding an order by pressing the enter or spacebar.
 */
html.other.add.focus()


html.add.table.appendChild(createTableOptionsHtml())
html.edit.table.appendChild(createTableOptionsHtml())
```
