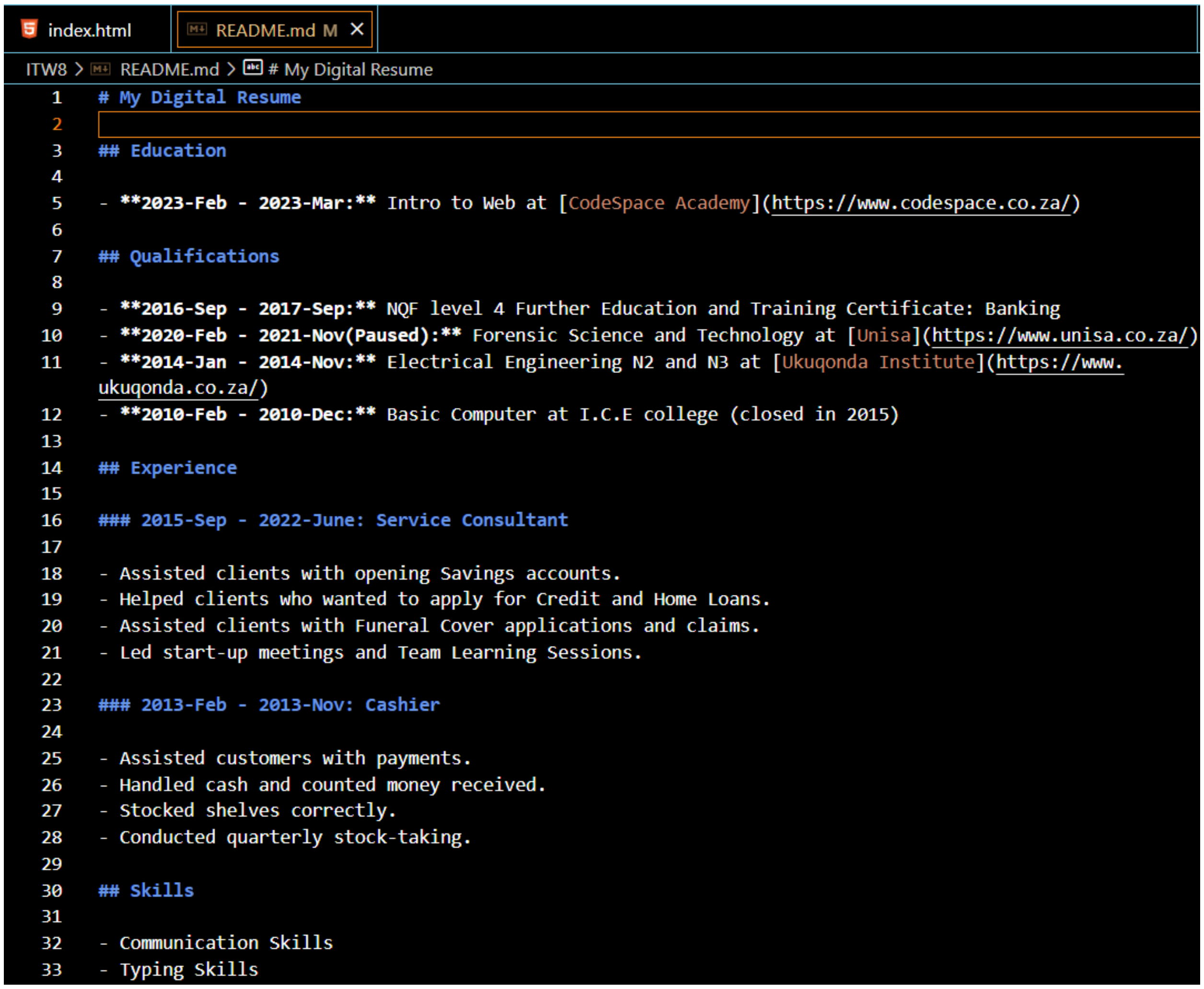
DWA_03.4 Knowledge Check_DWA3.1

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

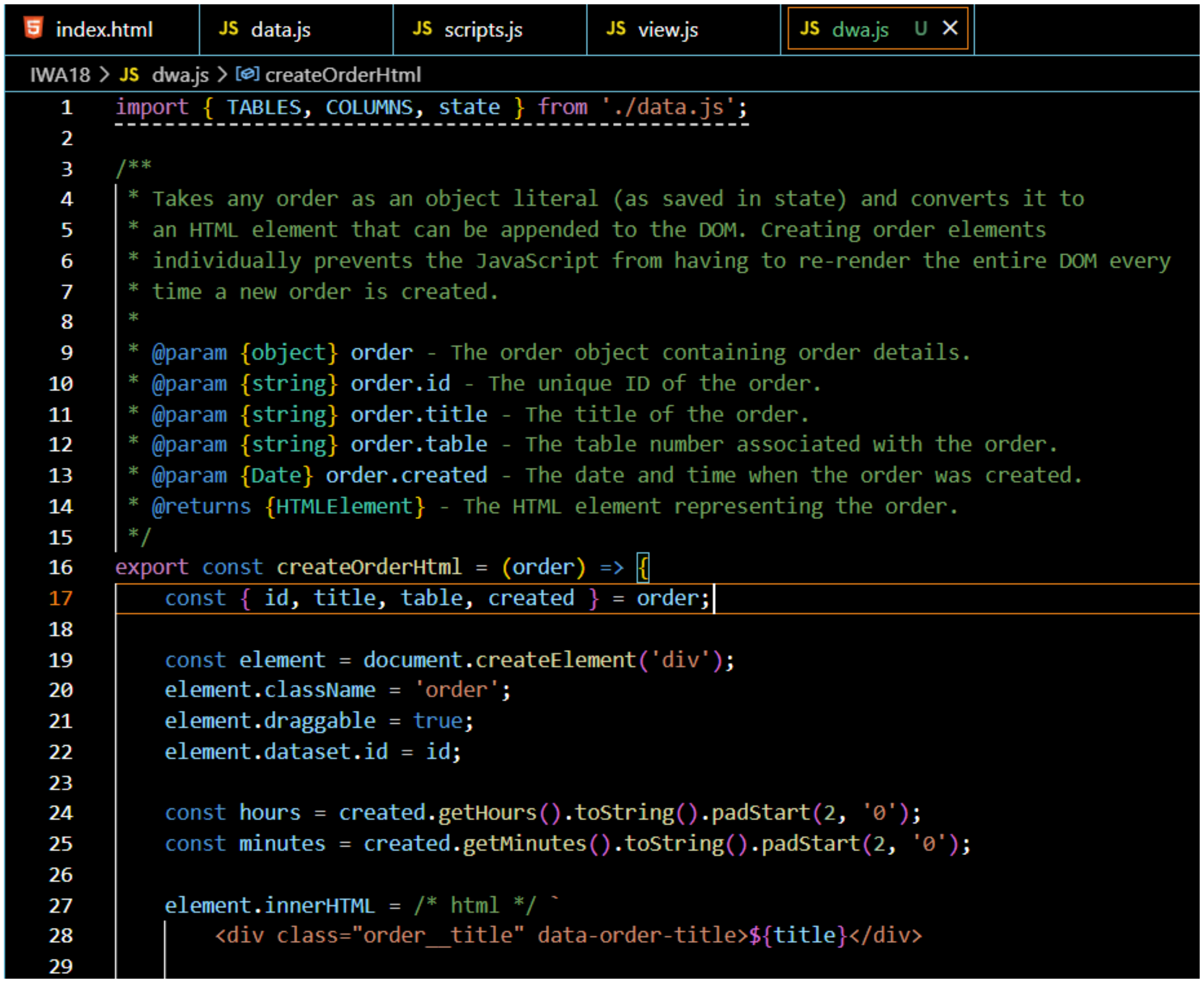


index.html | README.md M

ITW8 > README.md > # My Digital Resume

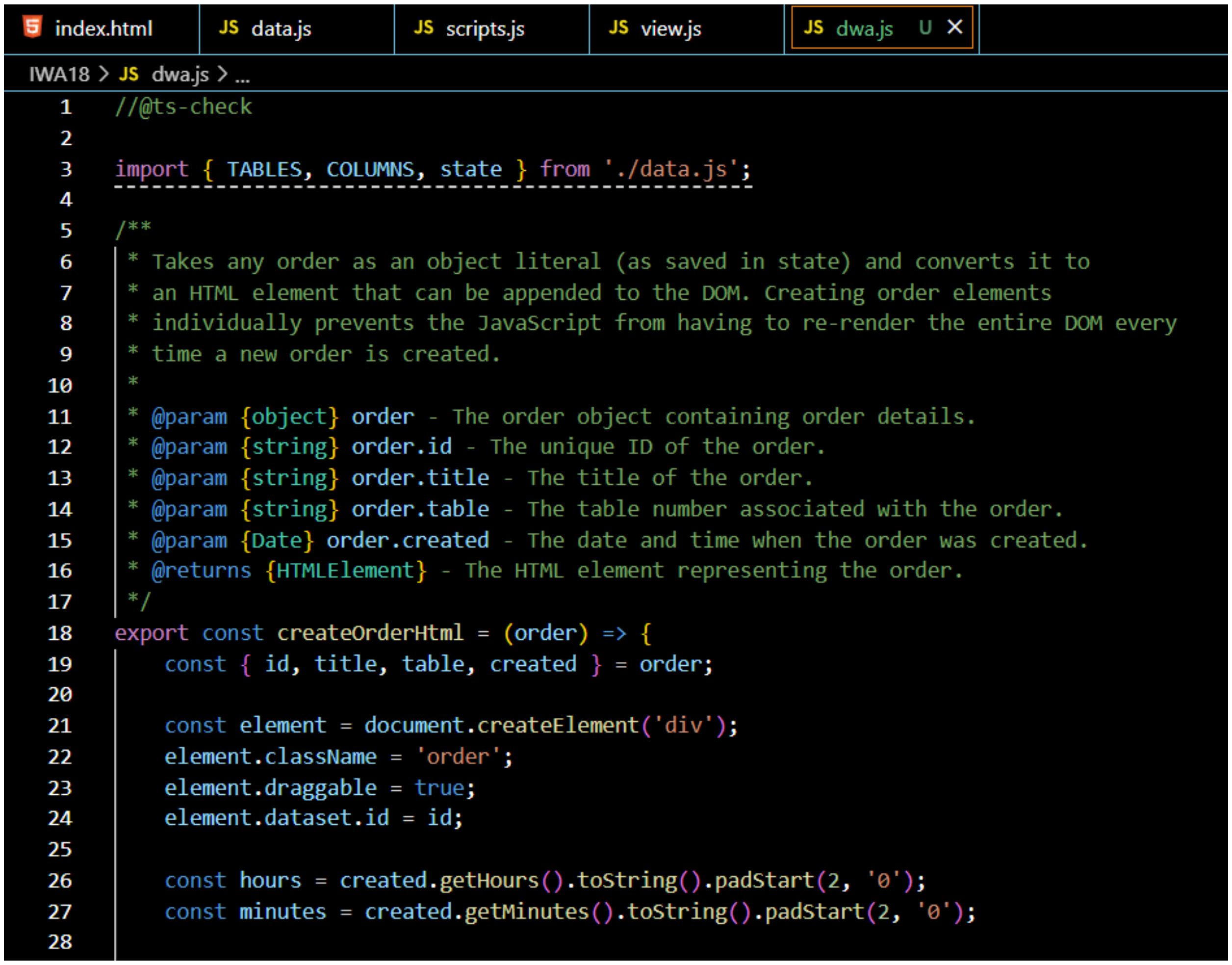
```
# My Digital Resume

## Education

- **2023-Feb - 2023-Mar:** Intro to Web at [CodeSpace Academy](https://www.codespace.co.za/)

## Qualifications

- **2016-Sep - 2017-Sep:** NQF level 4 Further Education and Training Certificate: Banking
- **2020-Feb - 2021-Nov(Paused):** Forensic Science and Technology at [Unisa](https://www.unisa.co.za/)
- **2014-Jan - 2014-Nov:** Electrical Engineering N2 and N3 at [Ukuqonda Institute](https://www.ukuqonda.co.za/)
- **2010-Feb - 2010-Dec:** Basic Computer at I.C.E college (closed in 2015)

## Experience

### 2015-Sep - 2022-June: Service Consultant

- Assisted clients with opening Savings accounts.
- Helped clients who wanted to apply for Credit and Home Loans.
- Assisted clients with Funeral Cover applications and claims.
- Led start-up meetings and Team Learning Sessions.

### 2013-Feb - 2013-Nov: Cashier

- Assisted customers with payments.
- Handled cash and counted money received.
- Stocked shelves correctly.
- Conducted quarterly stock-taking.

## Skills

- Communication Skills
- Typing Skills
```

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



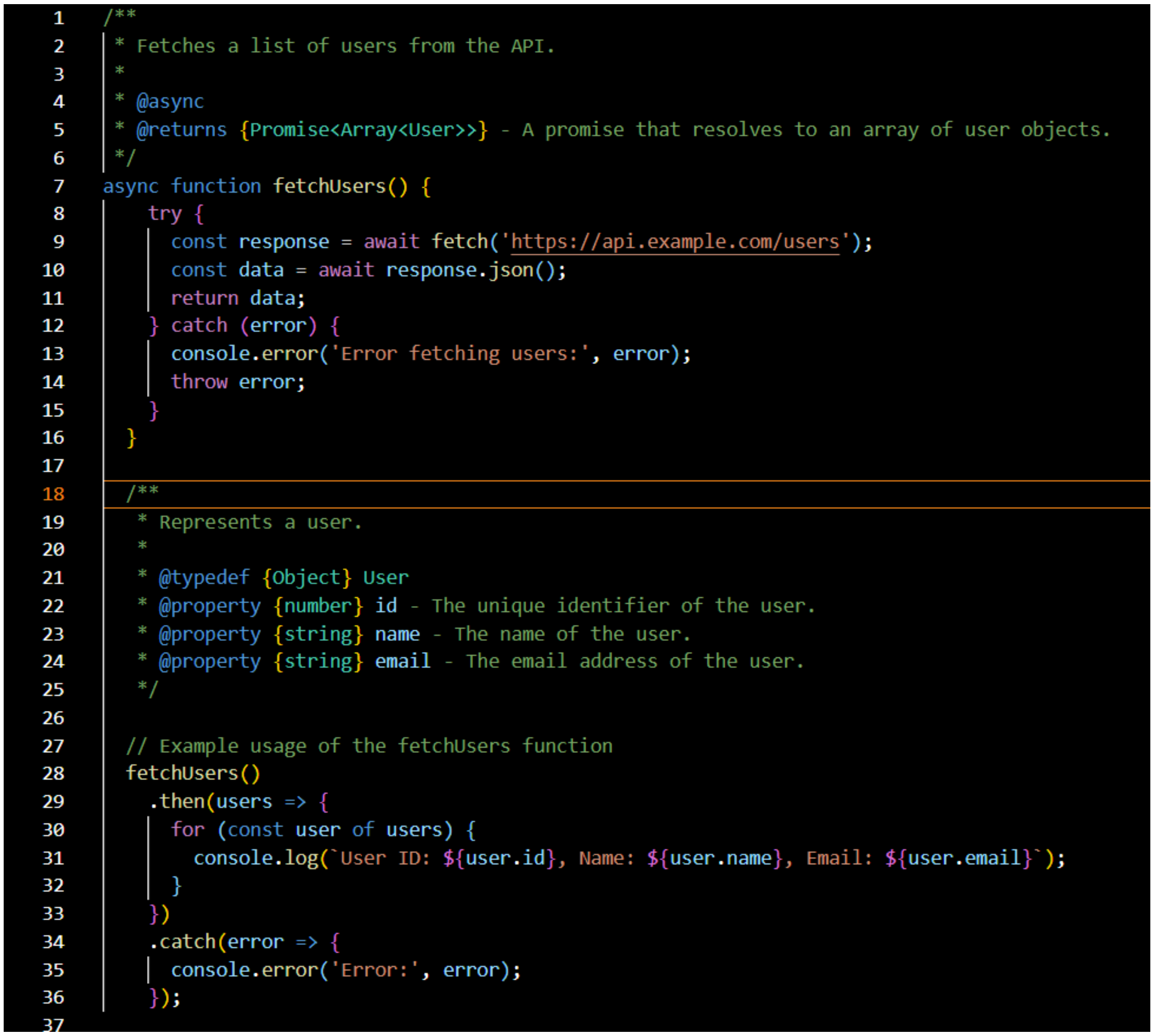
```javascript
import { TABLES, COLUMNS, state } from './data.js';

/**
 * Takes any order as an object literal (as saved in state) and converts it to
 * an HTML element that can be appended to the DOM. Creating order elements
 * individually prevents the JavaScript from having to re-render the entire DOM every
 * time a new order is created.
 *
 * @param {object} order - The order object containing order details.
 * @param {string} order.id - The unique ID of the order.
 * @param {string} order.title - The title of the order.
 * @param {string} order.table - The table number associated with the order.
 * @param {Date} order.created - The date and time when the order was created.
 * @returns {HTMLElement} - The HTML element representing the order.
 */
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.



```javascript
//@ts-check

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

/**
 * Takes any order as an object literal (as saved in state) and converts it to
 * an HTML element that can be appended to the DOM. Creating order elements
 * individually prevents the JavaScript from having to re-render the entire DOM every
 * time a new order is created.
 *
 * @param {object} order - The order object containing order details.
 * @param {string} order.id - The unique ID of the order.
 * @param {string} order.title - The title of the order.
 * @param {string} order.table - The table number associated with the order.
 * @param {Date} order.created - The date and time when the order was created.
 * @returns {HTMLElement} - The HTML element representing the order.
 */
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.



```javascript
/**
 * Fetches a list of users from the API.
 *
 * @async
 * @returns {Promise<Array<User>>} - A promise that resolves to an array of user objects.
 */
async function fetchUsers() {
    try {
      const response = await fetch('https://api.example.com/users');
      const data = await response.json();
      return data;
    } catch (error) {
      console.error('Error fetching users:', error);
      throw error;
    }
  }

  /**
   * Represents a user.
   *
   * @typedef {Object} User
   * @property {number} id - The unique identifier of the user.
   * @property {string} name - The name of the user.
   * @property {string} email - The email address of the user.
   */

  // Example usage of the fetchUsers function
  fetchUsers()
    .then(users => {
      for (const user of users) {
        console.log(`User ID: ${user.id}, Name: ${user.name}, Email: ${user.email}`);
      }
    })
    .catch(error => {
      console.error('Error:', error);
    });

```

I used the API documentation in the above code. It is just an example.