DWA_07.4 Knowledge Check_DWA7

---

1. Which were the three best abstractions, and why?

- Moved the querySelectors to their own folder to clear up and clean up my code from the main js folder.
- In this code below, I've divided the code into separate functions to improve readability and maintainability.

```
/**
 * This code sets up for a submit event listener. When the form is submitted, the selected
 * object is created by converting the form data to an object using Object.fromEntries().
 * Depending on the theme selected, the --color-light and --color-dark CSS variables are
 * updated with the corresponding light and dark color values from the css object
 */
dataSettingsForm.addEventListener('submit', handleFormSubmit);

export function handleFormSubmit(event) {
    event.preventDefault();
    const selected = getSelectedValues(event.target);

    if (selected.theme) {
        applyThemeStyles(selected.theme);
    }

    dataSettingsOverlay.close();
}

function getSelectedValues(form) {
    const formSubmit = new FormData(form);
    return Object.fromEntries(formSubmit);
}

function applyThemeStyles(theme) {
    if (css[theme] && css[theme].length >= 2) {
        document.documentElement.style.setProperty('--color-light', css[theme][0]);
        document.documentElement.style.setProperty('--color-dark', css[theme][1]);
    }
}
```

- Modularized the below code into smaller functions for creating elements with attributes and elements with text content. This approach enhances code readability, encourages code reuse.

```
function createPreview(preview) {
    const { author: authorId, id, image, title } = preview;

    const showPreview = createElementWithAttributes('button', {
        class: 'preview',
        'data-preview': id,
    });

    const imageElement = createElementWithAttributes('img', {
        class: 'preview__image',
        src: image,
    });

    const infoElement = createElementWithAttributes('div', {
        class: 'preview__info',
    });

    const titleElement = createElementWithText('h3', title);

    const authorElement = createElementWithText('div', authors[authorId]);

    infoElement.append(titleElement, authorElement);
    showPreview.append(imageElement, infoElement);

    return showPreview;
}

function createElementWithAttributes(tagName, attributes) {
    const element = document.createElement(tagName);
    for (const [attrName, attrValue] of Object.entries(attributes)) {
        element.setAttribute(attrName, attrValue);
    }
    return element;
}
```

---

2. Which were the three worst abstractions, and why?

- Commenting and Documentation.

---

3. How can The three worst abstractions be improved via SOLID principles.

- Below is an example of how the code could've been properly documented.

  Class level Documentation for the books.

```
/**
 * Represents a Book object.
 *
 * @class
 */
class Book {
    /**
     * Create a new Book instance.
     *
     * @constructor
     * @param {string} title - The title of the book.
     * @param {string} author - The author of the book.
     */
    constructor(title, author) {
        // Implementation details...
    }

    // Other methods...
}
```

  Function level documentation for the functions.

```
/**
 * Calculates the average of an array of numbers.
 *
 * @param {number[]} numbers - An array of numbers.
 * @returns {number} The average of the numbers.
 */
function calculateAverage(numbers) {
    // Implementation details...
}
```