DWA_08 Discussion Questions

In this module you will continue with your “Book Connect” codebase, and further iterate on your abstractions. You will be required to create an encapsulated abstraction of the book preview by means of a single factory function. If you are up for it you can also encapsulate other aspects of the app into their own abstractions.

---

1. What parts of encapsulating your logic were easy?

This is the book preview code :

```
//@ts-check

/**
 * Creates a preview element for a book.
 *
 * @param {Object} bookData - The data of the book for which to create the preview.
 * @param {Object} authors - An object containing author information with author IDs as keys and author names as values.
 * @returns {HTMLElement} The created preview element.
 */
export function createPreview(bookData, authors) {
    // Destructure bookData
    const { author, id, image, title, genre } = bookData;

    // Create the main preview element
    const preview = document.createElement('div');
    preview.classList.add('preview');
    preview.setAttribute('data-preview', id);

    // Create the book image element
    const imageElement = document.createElement('img');
    imageElement.classList.add('preview__image');
    imageElement.src = image;

    // Create the book information container
    const info = document.createElement('div');
```

```
    info.classList.add('preview__info');

    // Create the book title element
    const titleElement = document.createElement('h3');
    titleElement.classList.add('preview__title');
    titleElement.textContent = title;

    // Create the book genre element
    const genreElement = document.createElement('div');
    genreElement.classList.add('preview__genre');
    genreElement.textContent = genre;

    // Get the author's name from the authors object
    const authorName = authors[author];

    // Create the book author element
    const authorElement = document.createElement('div');
    authorElement.classList.add('preview__author');
    authorElement.textContent = authorName;

    // Append elements to the info container
    info.appendChild(genreElement);
    info.appendChild(titleElement);
    info.appendChild(authorElement);

    // Append elements to the main preview element
    preview.appendChild(imageElement);
    preview.appendChild(info);

    return preview;
}
```

Creating Elements: The code uses document.createElement() to create HTML elements like div, img, and h3. This is similar to building with building blocks. Creating elements like this is a straightforward process and doesn't involve complex concepts.

Extracting data from the bookData and authors objects using property names (like author, image, etc.) is like picking items from a list. It's a basic operation that doesn't involve complex calculations.

Creating Hierarchy: Putting elements inside other elements using appendChild() is similar to organizing things in folders. You're arranging them in a structured manner, and this hierarchy is easy to visualize.

---

2. What parts of encapsulating your logic were hard?

Understanding Function Structure: At first, understanding how to structure the function and where each step fits was a bit confusing. Deciding what goes inside the function and what stays outside.
Working with DOM: Manipulating the Document Object Model (DOM) using methods like createElement() and appendChild() felt intricate. Grasping how these methods interact to create the desired structure took some time.
Handling Complex Relationships: Dealing with relationships between various data points, like associating authors with their books, and dynamically creating elements based on this data.

---

3. Is abstracting the book preview a good or bad idea? Why?

Yes, code reusability, which enables reusing the logic across the application, avoiding duplication. Modular design, which creates an organized structure. Separation of concerns, keeping different aspects of code separate for easier maintenance.Code readability, enhances the understanding of complex logic.