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.

To prepare for your session with your coach, please answer the following questions. Then download this document as a PDF and include it in the repository with your code.

---

1. What parts of encapsulating your logic were easy?

Creating the BookPreview class and encapsulating the function so that it creates a book preview element, extracts necessary data from the book object and creates a button element for the preview and sets the inner HTML of the button with book data.

---

2. What parts of encapsulating your logic were hard?

Determining which exact functionalities to encapsulate and
Using this. And the get method

---

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

Encapsulating and abstracting the book preview is a good idea for several reasons:

- Modularity and Reusability : Encapsulating the functionality within a class promotes modularity and reusability. You can create multiple instances of the `BookPreview` class for different books, and each instance encapsulates the necessary logic and DOM manipulation for that specific book preview. This modular approach allows for easier maintenance, testing, and extensibility of the code.

- Abstraction and Encapsulation : By encapsulating the book preview functionality within a class, you can abstract away the implementation details and provide a clean interface to interact with book previews. Other parts of the code can interact with the `BookPreview` instances without needing to know the specific implementation details, promoting encapsulation and reducing coupling between components.

- Code Organization : Using a class helps in organizing related code together, making it easier to understand and navigate. All the logic and DOM manipulation related to book previews are encapsulated within the class, making it clear where to find and modify that functionality.

- Code Maintainability : Encapsulating functionality within a class improves code maintainability. If you need to make changes or add new features to the book preview functionality, you can do so within the class, keeping the changes contained and minimizing the impact on other parts of the codebase.

- Code Consistency : By encapsulating the functionality within a class, you establish a consistent and standardized way of creating and manipulating book previews. This consistency can help in reducing bugs and making the codebase easier to understand and maintain for multiple developers working on the project.
