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?

- Reusability
The factory function for creating book previews could be easily reused in multiple parts of the application.

- Separation of Concerns
Encapsulating the book preview creation in a factory function allowed for clear separation of concerns. The logic for constructing a book preview was isolated from other parts of the application, making the code more modular and easier to understand.

-Maintainability
By encapsulating the book preview creation, it became easier to maintain and make changes to the rendering of book previews. If changes were needed, you could modify the factory function without affecting other parts of the application.

---

2. What parts of encapsulating your logic were hard?

- Handling User Interactions
While encapsulating the creation of book previews is straightforward, handling user interactions with the book previews, such as clicking on them to display details, could introduce some complexity.

- Managing State

The management of the ‘page’ and ‘matches’ variables is crucial for controlling the state of the booklist.

- Documentation
Effective documentation becomes crucial when code is encapsulated.

---

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

Abstracting the book preview is a beneficial practice as it promotes modularity, reusability, readability, and consistency in the codebase. It simplifies maintenance, enables testing, and aligns with the separation of concerns principle. However, it’s crucial to strike a balance between abstraction and simplicity to avoid over-complexity in straightforward code.
