ZDWA_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?

Extracting the code into a function: The existing code was already contained within a function, createBookElement. It was a matter of renaming the function to createBookPreview and ensuring that it returns the generated element. This step required minimal modifications and made the encapsulation process relatively simple.

Maintaining compatibility: The goal was to encapsulate the logic without introducing breaking changes or altering the original behavior. By keeping the code structure and logic intact, and only making necessary adjustments for the function name and return statement, it was possible to maintain compatibility with the existing codebase.

Testing the encapsulated logic: Since the logic was already implemented, it was relatively easy to verify the encapsulated function's correctness by comparing its output with the existing code's output. Testing consisted of generating book previews using both approaches and ensuring they produce the same results.

---

2. What parts of encapsulating your logic were hard?

Ensuring modularity: One challenge could be ensuring that the encapsulated logic remains modular and independent. It's important to avoid tightly coupling the encapsulated function with other parts of the code. This can be achieved by keeping the

function self-contained and not relying on external variables or dependencies that may introduce dependencies or side effects.

Managing complex or convoluted code: the original code was complex or convoluted, encapsulating the logic could be more challenging. In such cases, it might be necessary to refactor or simplify the code before encapsulation to ensure the encapsulated function is clear, maintainable, and easier to understand.

---

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

Abstracting the book preview is generally a good idea for several reasons:

Code Readability: By encapsulating the logic for creating the book preview in a separate function, the code becomes more readable and easier to understand. The function name, createBookPreview, clearly communicates its purpose, making the code more self-explanatory.

Reusability: Abstracting the book preview logic into a function promotes code reusability. Once encapsulated, the function can be easily reused whenever a book preview element needs to be created, reducing code duplication and improving maintainability.
