DWA08 Challenge

Question 1:
What problems did you encounter converting the book preview to a component?

CSS Styling: I adapted existing CSS classes like "preview" for my web component, making adjustments for global and conflicting styles within the app.

Data Binding: Adapting my function for a web component meant managing data internally via attributes or properties within the component, rather than relying on external arguments.

Event Handling: Creating a book preview component with interactive features required careful attachment and management of event listeners within the component's logic.

Reuse and Maintenance: I structured the web component to enable easy reuse across the app without conflicts, aligning with the reusability goal of web components.

Component Lifecycle: Adapting to changing book or author data required handling dynamic updates and rendering adjustments, adding complexity to the conversion process.

Question 2:
What other elements make sense to convert into web components? Why?

Custom UI Elements: Elements like modals, tooltips, sliders, and tabs can be encapsulated as web components. This promotes modular and reusable code.

Widgets: Any interactive widgets like date pickers, color pickers, or charts can be turned into web components for easy integration across different projects.

Complex Components: Components that have their own internal logic and rendering, like a collapsible tree view or an image gallery, benefit from being isolated as web components.

UI Patterns: If your application uses certain UI patterns repeatedly, creating web components for them ensures consistent implementation and easier maintenance.

Third-Party Integrations: If you're using third-party libraries or UI kits, wrapping them in web components can help abstract away their complexity and prevent global namespace pollution.

Question 3:
Why does keeping your HTML, CSS, and JavaScript in a single file sometimes make sense?

Small Projects: For small projects or quick prototypes, having everything in one file can simplify the development process and reduce the overhead of managing multiple files.

Learning and Examples: When teaching or providing examples, a single file can make it easier for learners to see the entire code in one place, aiding comprehension.

Isolated Components: If the component's code is relatively short and self-contained, having it all in one place can make it more portable and easier to share.

Quick Edits: When making minor changes or quick edits, it's faster to locate and modify code within a single file.