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

- When converting the book preview into a component, you may need to adjust the existing CSS styles used in the preview. This is important to make sure that the component's styling stays separate from other parts of the application and doesn't create conflicts.
- Scoped CSS: Keep the component's styles contained within the component itself, so they only affect the component's elements and not other parts of the application.
- Class names: Choose specific class names for the component's styles that won't clash with other class names in the application. Consider adding a unique identifier to the class names to avoid conflicts.

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

To determine what other elements could be converted into web components, we need to consider the reusability and encapsulation benefits that web components offer. Web components are self-contained and reusable elements that can be used across different parts of an application without causing conflicts.

- Search Overlay: The search overlay functionality seems to be used to display a search form and handle search-related actions. It could be converted into a web component to encapsulate the search functionality and allow it to be easily reused in other parts of the application.
- Settings Overlay: Similar to the search overlay, the settings overlay could be converted into a web component to encapsulate the settings functionality and make it reusable.
- Option Elements: The createOptionElements function generates option elements for dropdown menus. It could be converted into a web component that accepts the container name, default value, and options as properties, making it easier to create dropdown menus with different options throughout the application.
- List Items: The list items generated in the createBookPreview function could be encapsulated within a web component, allowing for easier management and customization of book previews in different parts of the application.
- Theme Switcher: The code includes a theme switcher functionality that modifies CSS variables based on the selected theme. This could be encapsulated within a web component, making it easier to manage and reuse the theme switcher across different sections of the application.
- Load More Button: The load more button functionality could be encapsulated within a web component, allowing for consistent behaviour and easier reuse in other parts of the application that require a similar load more functionality.

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

Keeping HTML, CSS, and JavaScript in a single file, also known as inline coding or inline styling, can sometimes make sense in certain scenarios. Here are a few reasons why it might be beneficial:

- Simplicity and Ease of Deployment: Combining all code into a single file simplifies the deployment process. It reduces the number of files that need to be managed and uploaded, making it easier to distribute the code or share it with others. This can be particularly useful for small projects or quick prototypes where simplicity and speed are prioritised over code organisation.
- Rapid Development and Iteration: During the early stages of development or when rapidly prototyping, having all the code in one file can speed up the development process. It eliminates the need to switch between multiple files and reduces the overhead of managing separate files. This approach allows developers to quickly experiment, make changes, and iterate on the code.
- Small-scale Projects: For small-scale projects with limited functionality and a small codebase, combining HTML, CSS, and JavaScript into a single file can be a pragmatic choice. It keeps the codebase compact and self-contained, making it easier to understand and maintain.
- Code Portability: In some cases, such as creating code snippets or embedding code in specific environments (e.g., content management systems or email templates), having a single file can enhance code portability. It ensures that all necessary code is bundled together and can be easily copied and pasted into different contexts without missing dependencies.
- Performance Optimization: Inline coding can potentially improve performance in certain scenarios. When the code is inlined, the browser doesn't need to make separate HTTP requests for additional files, reducing network latency. However, this performance benefit is typically minor and might be outweighed by other considerations in larger projects.