DWA_09 Discussion Questions

In this challenge, you will continue with your “Book Connect” codebase and further iterate on your abstractions.

Previously, you worked on adding abstraction around the book preview functionality of the project. Next, you must turn the book preview abstraction into a fully-working web component. Then, apply the techniques you’ve learned about this module to the book preview.

---

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

```
// Define a custom element for the book preview
window.customElements.define('book-preview', class BookPreview extends
HTMLElement {
    constructor() {
        super();

        // Extract data from attributes
        const id = this.getAttribute('data-id');
        const image = this.getAttribute('data-image');
        const title = this.getAttribute('data-title');
        const genre = this.getAttribute('data-genre');
        const author = this.getAttribute('data-author');

        // Create the main preview element
        const preview = document.createElement('div');
        preview.classList.add('preview');
        preview.setAttribute('data-preview', id);

        // Create the book image element
        const imageElement = document.createElement('img');
        imageElement.classList.add('preview__image');
        imageElement.src = image;

        // Create the book information container
        const info = document.createElement('div');
        info.classList.add('preview__info');

        // Create the book title element
        const titleElement = document.createElement('h3');
        titleElement.classList.add('preview__title');
        titleElement.textContent = title;

        // Create the book genre element
        const genreElement = document.createElement('div');
        genreElement.classList.add('preview__genre');
        genreElement.textContent = genre;

        // Create the book author element
        const authorElement = document.createElement('div');
        authorElement.classList.add('preview__author');
        authorElement.textContent = author;

        // Append elements to the info container
        info.appendChild(genreElement);
        info.appendChild(titleElement);
        info.appendChild(authorElement);

        // Append elements to the main preview element
        preview.appendChild(imageElement);
        preview.appendChild(info);

        // Append the preview to the web component
        this.appendChild(preview);
    }
});
```

During the conversion of the book preview into a web component, there were a couple of challenges. First, we needed to extract essential data such as the book's title, author, and image from attributes in the HTML element. While this seemed straightforward, any missing or incorrect attributes could potentially lead to issues. Another concern was styling the component, as we defined its styles using class names, but we don't know how these styles will work with the rest of the webpage.

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

Theme Switcher: This is like a tool we use everywhere in our app to change its look. Making a "theme-switcher" component means we build it once and use it everywhere, so our app looks the same no matter where we are.

"Show More" Button: We often need a button that lets users see more stuff in our app. A "show-more-button" component makes it easy. We create it once, and we can use it everywhere in our app without doing the same work over and over.

Search Overlay: When we have a search box that pops up in the same way all over our app, a "search-overlay" component is handy. It takes care of how the search box appears and disappears, saving us from writing the same code each time.

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

From my understanding, having everything in one file can simplify things, especially for smaller projects. It's easier to distribute and share. But I'm also aware of the downsides. As projects get bigger, maintaining everything in a single file might become challenging. Separating concerns between HTML, CSS, and JavaScript is a good practice for readability.