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?

Encapsulating can be defined as the practice of bundling data and the methods that operate on that data into an object, or single unit. Using the try/catch statements were fairly easy, though I did struggle a bit. With more practice it comes easier though. I also made use of functions, like in line 104 of the list button.
The reason why I used the try/catch method is because I wanted to encapsulate potential errors in order to isolate the error handling logic. I believe that by continuing to work more with encapsulating, it will become more intuitive and will continue making error handling easier.

```
87  //used the try/catch method
88  function setTheme() {
89      try {
90          if (window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches) {
91              document.querySelector('[data-settings-theme]').value = 'night';
92              document.documentElement.style.setProperty('--color-dark', '255, 255, 255');
93              document.documentElement.style.setProperty('--color-light', '10, 10, 20');
94          } else {
95              document.querySelector('[data-settings-theme]').value = 'day';
96              document.documentElement.style.setProperty('--color-dark', '10, 10, 20');
97              document.documentElement.style.setProperty('--color-light', '255, 255, 255');
98          }
99      } catch (error) {
00          console.error('Error setting theme:', error);
01      }
02  }
03   //used function as part of my abstraction
04  function updateListButton() {
05      try {
06          const listButton = document.querySelector('[data-list-button]');
07          const remainingCount = Math.max(matches.length - (page * BOOKS_PER_PAGE), 0);
08          listButton.innerText = `Show more (${books.length - BOOKS_PER_PAGE})`;
09          listButton.disabled = remainingCount > 0;
10          listButton.innerHTML = `
11            <span>Show more</span>
12            <span class="list__remaining"> (${remainingCount})</span>
13          `;
14      } catch (error) {
15         console.error('Error updating list button:', error);
16      }
17  }
```

---

2. What parts of encapsulating your logic were hard?

I failed to use the try/catch statement at line 119 though, the end result kept on creating errors. Trying to add functions on it disabled my “Search button” and the “Night and Dark Modes”.
I think that one of my challenges was to identify the right abstraction. We are taught that there are Algorithms, Data structures, Modules, Classes, Frameworks and Functions. I tried to use mostly Functions and I think that could have contributed to making the logic hard.
Transitioning from the working code in DWA06 to having to edit it in DWA08 also showed me that there are so many different ways to abstract code, and write code as well.

---

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

For me personally, it's not a good idea. I realized that I tend to over-abstract and thereby increase the complexity of my code. I found it to be more challenging. I ended up finding it a bit more difficult to follow. I also feared leaks in my code due to abstracting.
