DWA_07.4 Knowledge Check_DWA7

---

1. Which were the three best abstractions, and why?

My bookMoreDetailsAbs.js I thought was a good abstraction because it was a short class that handled what happened when a book was clicked on and what displayed on the screen in a precise class specific for that function.

I also thought the bookFilterAbs.js was a good one because, although it was a long class, everything pertaining to filtering a book is in this one space. It handles the actual filter, the display of the filter and the click event when clicking on the filter button. If anything needs to be changed it can happen in that one js document.

I only had 5 main abstractions and changed 3 of them, so there were only 2 I thought were the best.

---

2. Which were the three worst abstractions, and why?

I didn’t necessarily think that any of my abstractions were “bad”, however, I did apply functional programming to 3 of them after completing DWA7.

The first one I changed was the themeHandler.js because this abstraction would work nicely broken up further into smaller functions. I made this greater function of the smaller ones that got the css theme, applied the theme to the document, and to choose what theme the user wanted.

Then, I changed the bookLoaderAbs.js because this was also a good one to make functional programming work for. This was also one long class in DWA6 which I broke into 4 functions- updating the actual show more button, checking the status of the button (if it should still be active or not), updating the remaining number of books and loading the next batch of books each time the button is clicked.

This led me to the booksDisplayAbs.js where this was one function that dealt with the displaying of the books. It’s very similar to the class I had written for this function in DWA6 but written differently and as a function rather than as a class.

---

3. How can The three worst abstractions be improved via SOLID principles.

Each of my abstractions only deal with ONE principle (SRP) or functionality eg. the themeHandler only deals with the theme and no other part of the program functioning. My abstractions all also apply the ISP so every method in the code is actually utilized and there is no excess code taking up space.
