Keotshepile Maje

DWA_07.4 Knowledge Check_DWA7

---

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

```
const htmlReference = {

    headerSearch: document.querySelector('[data-header-search]'),
    headerSettings: document.querySelector('[data-header-settings]'),

    listItems: document.querySelector('[data-list-items]'),
    listMessage: document.querySelector('[data-list-message]'),
    listButton: document.querySelector('[data-list-button]'),
    listActive: document.querySelector('[data-list-active]'),
    listBlur: document.querySelector('[data-list-blur]'),
    listImage: document.querySelector('[data-list-image]')
```

- htmlReference Object
  The htmlReference is contain all the references from HTML and made it easy to work with this reference as a variable over and over again whenever in need.

```
96  const showMoreButton = () => {
97
98      remainingBooks = (matches.length - (page * BOOKS_PER_PAGE))
99
100     listButton.innerHTML = `
101         <span>Show more</span>
102         <span class="list__remaining"> (${ remainingBooks > 0 ? remainingBooks : 0})</span>
103     `
104 }
105
```

.

The code can be used every time it's called and is able to update the remaining Book length without giving incorrect number or calculation.

```
 * @param {string} - the parameter should only be 'genres' or 'authors'
 */

let createOption = (param) => {
    const firstElement = document.createElement('option')
    firstElement.value = 'any'
    firstElement.innerText = 'All Genres'
    optionFragment.appendChild(firstElement)

    for (const [id, name] of Object.entries(param)) {
        const element = document.createElement('option')
        element.value = id
        element.innerText = name
        optionFragment.appendChild(element)
    }

}

//Call the function to create options for genres
createOption(genres)
searchGenres.appendChild(optionFragment)

/**
 * The genres data, from data.js is added on the search form for users to select the the books with only the genre they like.
 */
export const optionsForGenres = searchGenres.appendChild(optionFragment)

//Call the function to create options for authors
createOption(authors)
searchAuthors.appendChild(optionFragment)

/**
 * The author data, from data.js is added on the search form for users to select the the books with only the author they like.
 */
export const optionsForAuthors = searchAuthors.appendChild(optionFragment)
```

- createOptions function
  I was able to transfer this code to a different js module file that deals with DOM and it didn't give me an issue. It works as it was supposed to work with no problems.

---

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

- displayBooks() function
  The displayBooks function has more than one responsibility. It creates DOM and also loops over everything making it a bit complex. And when this function is used from a different module. It created bugs in the main script and results in undesired behavior.

---

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

Using the SOLID method I can make sure that apply the S- Single Responsibility principle. Where each abstraction is responsible for doing only one function.
