# DWA_07.4 Knowledge Check_DWA7

---

1. Which were the three best abstractions, and why?
    1. This is a good abstraction because it is a simple function that handles only one task. It would have been better to call the function with “day” or “night” as a parameter that way you can put checks in place that it wont be called if html.settings.theme.value = ‘night..

```
/**
 * Handles the saving of settings and updates the CSS styling
 * based on the Day/Night option value.
 * @param {Event} event - The event object.
 */
const handleSettingsSave = (event) => {
    event.preventDefault();
    if ((html.settings.theme.value = "night")) {
        document.documentElement.style.setProperty("--color-dark", "255, 255, 255");
        document.documentElement.style.setProperty("--color-light", "10, 10, 20");
    } else {
        document.documentElement.style.setProperty("--color-dark", "10, 10, 20");
        document.documentElement.style.setProperty(
            "--color-light",
            "255, 255, 255"
        );
    }
    html.settings.dialog.removeAttribute("open");
};

html.settings.save.addEventListener("click", handleSettingsSave);

```

    2. 

```
// this appends the Genre <options> to the search dropdownlist
const createGenreOptionsHtml = () => {
    const fragment = document.createDocumentFragment();

    for (const [key, value] of Object.entries(genres)) {
        const option = document.createElement("option");
        option.value = key;
        option.innerText = value;
        fragment.appendChild(option);
    }

    html.search.genres.appendChild(fragment);
};

// the fuction is called when the list is clicked
html.search.genres.addEventListener("click", createGenreOptionsHtml);

// the createAuthorOptionsHtml works in the same way as Genres. A future project will be to sort items
// so only relavant authors are shows after a genre is chosen or vice verca
const createAuthorOptionsHtml = () => {
    const fragment = document.createDocumentFragment();

    for (const [key, value] of Object.entries(authors)) {
        const option = document.createElement("option");
        option.value = key;
        option.innerText = value;
        fragment.appendChild(option);
    }
    html.search.authors.appendChild(fragment);
};
html.search.authors.addEventListener("click", createAuthorOptionsHtml);
```

    3. In third place in the best and worst list are functions handleSettingsToggle handleSearchToggle respectively. These Two are incredibly similar however they showcase a tendency in my previous code to group the HTML and the Javascript

functions together. There are worse offenders but they showcase how clearing HTML elements should be kept separate.  However both function toggle the window in a way that is elegant.

```
/**
 * Opens and closes the settings dialog.
 * @param {Event} event - The event object.
 */
const handleSettingsToggle = (event) => {
    event.preventDefault();
    if (html.settings.dialog.hasAttribute("open")) {
        html.settings.dialog.removeAttribute("open");
    } else {
        html.settings.dialog.setAttribute("open", true);
    }
};
// Linking the function to its respective buttons using event lisners
html.settings.button.addEventListener("click", handleSettingsToggle);
html.settings.cancel.addEventListener("click", handleSettingsToggle);
```

---

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

1. showSearchResults is too convoluted. Multiple things are happening and they pertain to different parts of the program. Interacting with the database , creating new objects, converting those objects into HTML and adding that HTML to the webpage.
   This also has the added convoluted function of having to display messages like ‘no books in your selection’ which should be kept totally spereate

```
const showSearchResults = (object) => {

    if (typeof object == 'undefined'){
        html.search.dialog.removeAttribute('open')
        return
    } else if (object.length === 0){
        area.innerHTML= "no books match your description"
    }else {

        area.innerHTML = "";
        const extracted = object.slice(index, index + BOOKS_PER_PAGE);

        for (let i = 0; i < object.length; i++) {
                const book = object[i];
                const image = book.image;
        const title = book.title;
        const authorId = book.author;
        const id = book.id;

        const element = document.createElement("button");
        element.classList = "preview";
        element.setAttribute("id", id);

        element.innerHTML = `<img class="preview__image" src=${image}>
                <div class="preview__info">
                    <h3 class="preview__title">${title}</h3>
                    <div class="preview__author">${authors[authorId]}</div>
                </div> `;
        fragment.appendChild(element);
    }

    index = index + extracted.length;
    area.appendChild(fragment);
    const booksLeft = Object.keys(books).length - index;
    //set books left for "show more" button
    html.list.button.innerHTML = `Show More (${booksLeft})`;
    html.search.dialog.removeAttribute("open");
};
```

2. For the same reasons as the previous function but it is not as bad because it does not display messages.

```
// a function that loads a certain amount of books to the webpage everytime it is called
const showBooks = (object) => {
    const extracted = books.slice(index, index + BOOKS_PER_PAGE);

    for (let i = index; i < index + BOOKS_PER_PAGE; i++) {
        const book = books[i];
        const image = book.image;
        const title = book.title;
        const authorId = book.author;
        const id = book.id;

        const element = document.createElement("button");
        element.classList = "preview";
        element.setAttribute("id", id);

        element.innerHTML = `<img class="preview__image" src=${image}>
                <div class="preview__info">
                    <h3 class="preview__title">${title}</h3>
                    <div class="preview__author">${authors[authorId]}</div>
                </div> `;
        fragment.appendChild(element);
    }

    index = index + extracted.length;
    area.appendChild(fragment);
    const booksLeft = Object.keys(books).length - index;
    //set books left for "show more" button
    html.list.button.innerHTML = `Show More (${booksLeft})`;
};
```

3. See 1.3 for explanation of both

```
/**
 * Opens and closes the search dialog and clears the elements after use.
 * @param {Event} event - The event object.
 */
const handleSearchToggle = (event) => {
    event.preventDefault();
    html.search.title.value = "";
    html.search.genres.value = "";
    html.search.authors.value = "";

    if (html.search.dialog.hasAttribute("open")) {
        html.search.dialog.removeAttribute("open");
        // Remove all genre options except the first one
        html.search.genres
            .querySelectorAll("option:not(:first-child)")
            .forEach((option) => option.remove());
        // Remove all author options except the first one
        html.search.authors
            .querySelectorAll("option:not(:first-child)")
            .forEach((option) => option.remove());
    } else {
        html.search.dialog.setAttribute("open", true);
    }
};
```

---

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

This code does not utilize inheritance or extension of existing classes , dependencies or interfaces , so the  Open/Closed Principle (OCP), Liskov Substitution Principle (LSP), Interface Segregation Principle (ISP), and Dependency Inversion Principle (DIP) are not violated.

The bulk of my issues come in the form of the Single Responsibility Principle (SRP).
In the case of option 1 and 2, showBooks and showSearchResults, they clear the HTML, handle the data from the database, use that data to create HTML and then add that HTML to the webpage.
If i did this again i would:
Create a function that clears the HTML elements on the page
Create a function that (given an object that contains all the books (and data) it must display as a  parameter) turns that into HTML and adds the HTML to the page. This can be called:
When the page fist loads,
After the search function is called
When the display more button is clicked
A function that takes the search elements as parameters and filters the database creating a new object of the matches
A function that extracts the first BOOKS_PER_PAGE amount from an object

---