DWA_07.4 Knowledge Check_DWA7

---

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

```
**
 * @typedef {Object} selectors - html element data attribute values

 *
 */

/**
 * Html object containing Html data Attributes
 * @type {selectors}
 */
const selectors = {
  dataListItems: document.querySelector("[data-list-items]"),
  dataSearchGenres: document.querySelector("[data-search-genres]"),
  dataSearchAuthors: document.querySelector("[data-search-authors]"),
  dataSettingTheme: document.querySelector("[data-settings-theme]"),
  dataListButton: document.querySelector("[data-list-button]"),
  dataSearchCancel: document.querySelector("[data-search-cancel]"),
  dataSettingsCancel: document.querySelector("[data-settings-cancel]"),
  dataSearchOverlay: document.querySelector("[data-search-overlay]"),
  dataSettingsOverlay: document.querySelector("[data-settings-overlay]"),
  dataHeaderSearch: document.querySelector("[data-header-search]"),
  dataSearchtTitle: document.querySelector("[data-search-title]"),
  dataHeaderSettings: document.querySelector("[data-header-settings]"),
  dataListClose: document.querySelector("[data-list-close]"),
  dataSettingsForm: document.querySelector("[data-settings-form]"),
  dataListActive: document.querySelector("[data-list-active]"),
  dataListMessage: document.querySelector("[data-list-message]"),
  dataSearchForm: document.querySelector("[data-search-form]"),
  dataListBlur: document.querySelector("[data-list-blur]"),
  dataListImage: document.querySelector("[data-list-image]"),
  dataListTitle: document.querySelector("[data-list-title]"),
  dataListSubtitle: document.querySelector("[data-list-subtitle]"),
  dataListDiscription: document.querySelector("[data-list-description]"),
};
```

In this abstraction all the querySelectors have been grouped up so they can be accessed easier.

```
/**
 * Default html for preview
 * @param {string} picture
 * @param {string} heading
 * @param {Object} object
 * @param {string} property
 * @returns {string}
 */
const previewHtml = (picture, heading, object, property ) => {
  return `
  <img
      class="preview__image"
      src="${picture}"
  />

  <div class="preview__info">
      <h3 class="preview__title">${heading}</h3>
      <div class="preview__author">${object[property]}</div>
  </div>
  `
}
```

In this abstraction the html was used 3 times and using this abstraction makes it easier to read and not having to make unnecessary mistakes.

---

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

```
// Generate new book previews for the next page
for (const { author, id, image, title } of matches.slice(
  page * BOOKS_PER_PAGE,
  (page + 1) * BOOKS_PER_PAGE
)) {
  const element = document.createElement("button");
  // @ts-ignore
  element.classList = "preview";
  element.setAttribute("data-preview", id);

  element.innerHTML = previewHtml(image, title, authors, author)

  fragment.appendChild(element);
}
```

In this for of loop the calculation done after the match.slice can be made into a separate function as writing it like this makes it more complex.

---

3. How can The three worst abstractions be improved via SOLID principles.
-In this for of loop the calculation done after the match.slice can be made into a separate function as writing it like this makes it more complex.
-
