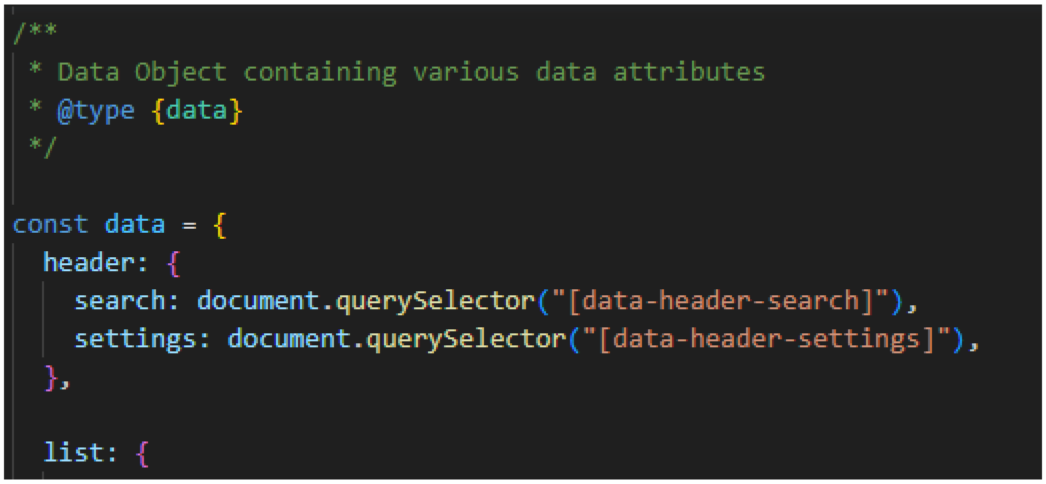
DWA_07.4 Knowledge Check

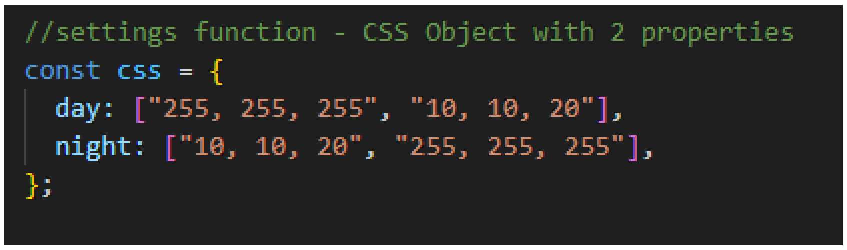
1. Which were the three best abstractions, and why?

    - Data Objects for DOM Elements – Abstracts the process of creating HTML elements, Organizes sections & headings for easier accessibility & better readability.



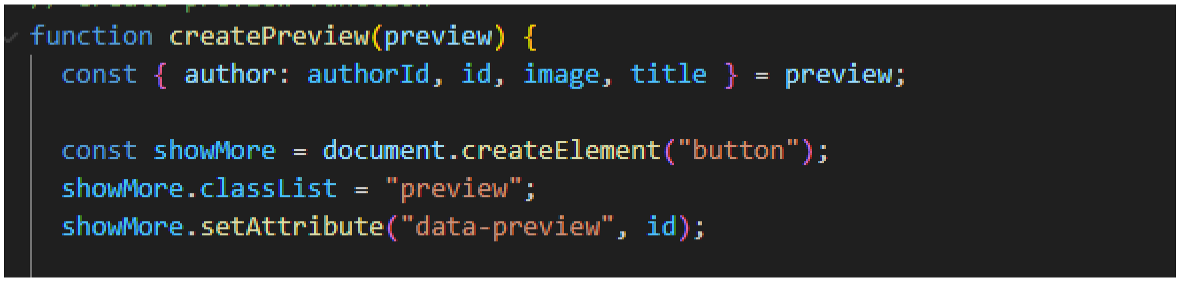
```
/**
 * Data Object containing various data attributes
 * @type {data}
 */

const data = {
  header: {
    search: document.querySelector("[data-header-search]"),
    settings: document.querySelector("[data-header-settings]"),
  },

  list: {
```

    - CSS Object for Theme Settings – Abstracts the process for switching between themes.



```
//settings function - CSS Object with 2 properties
const css = {
  day: ["255, 255, 255", "10, 10, 20"],
  night: ["10, 10, 20", "255, 255, 255"],
};
```

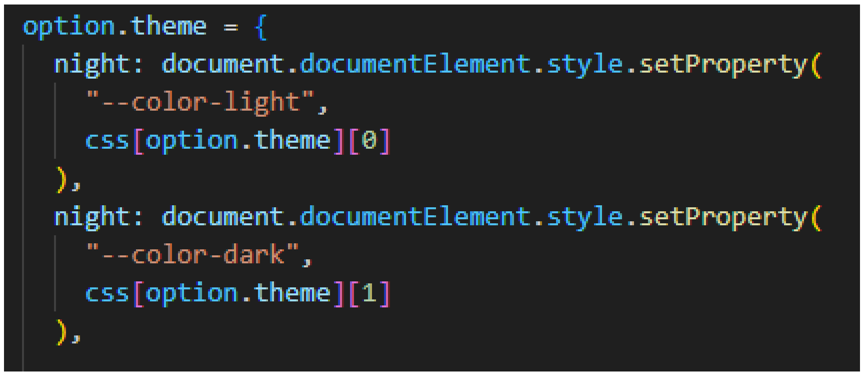
    - Create Preview Function - Abstracts the logic for opening the preview and retrieving the data.



```
function createPreview(preview) {
  const { author: authorId, id, image, title } = preview;

  const showMore = document.createElement("button");
  showMore.classList = "preview";
  showMore.setAttribute("data-preview", id);
```

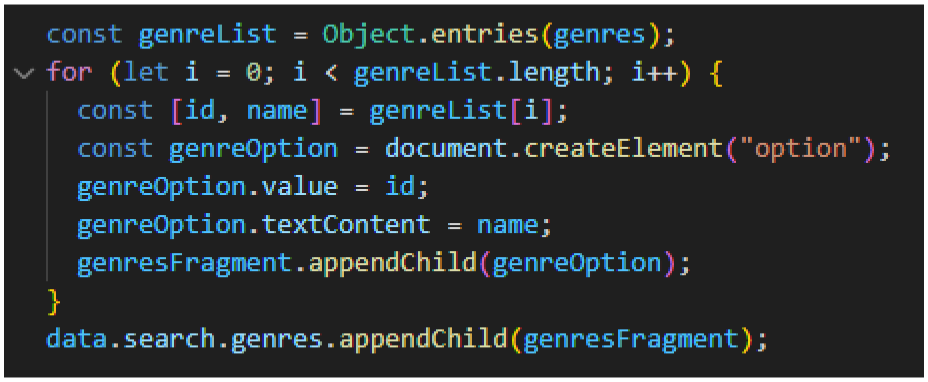
2. Which were the three worst abstractions, and why?

- Duplicate Key Values –

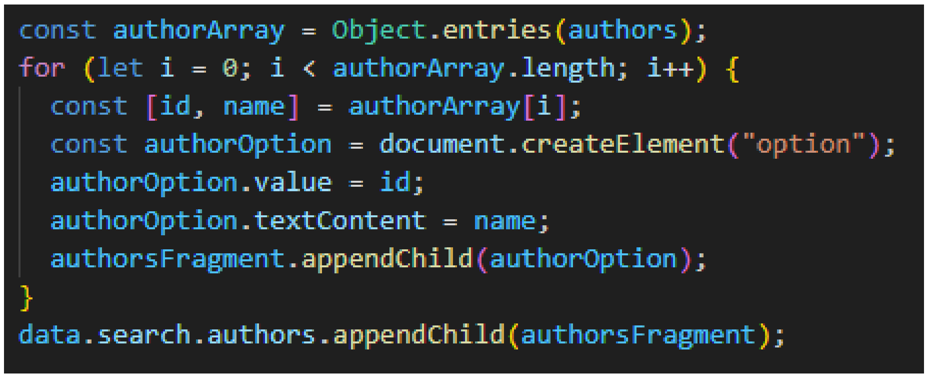


```
option.theme = {
  night: document.documentElement.style.setProperty(
    "--color-light",
    css[option.theme][0]
  ),
  night: document.documentElement.style.setProperty(
    "--color-dark",
    css[option.theme][1]
  ),
```

- Duplicate fragments with the same purpose -



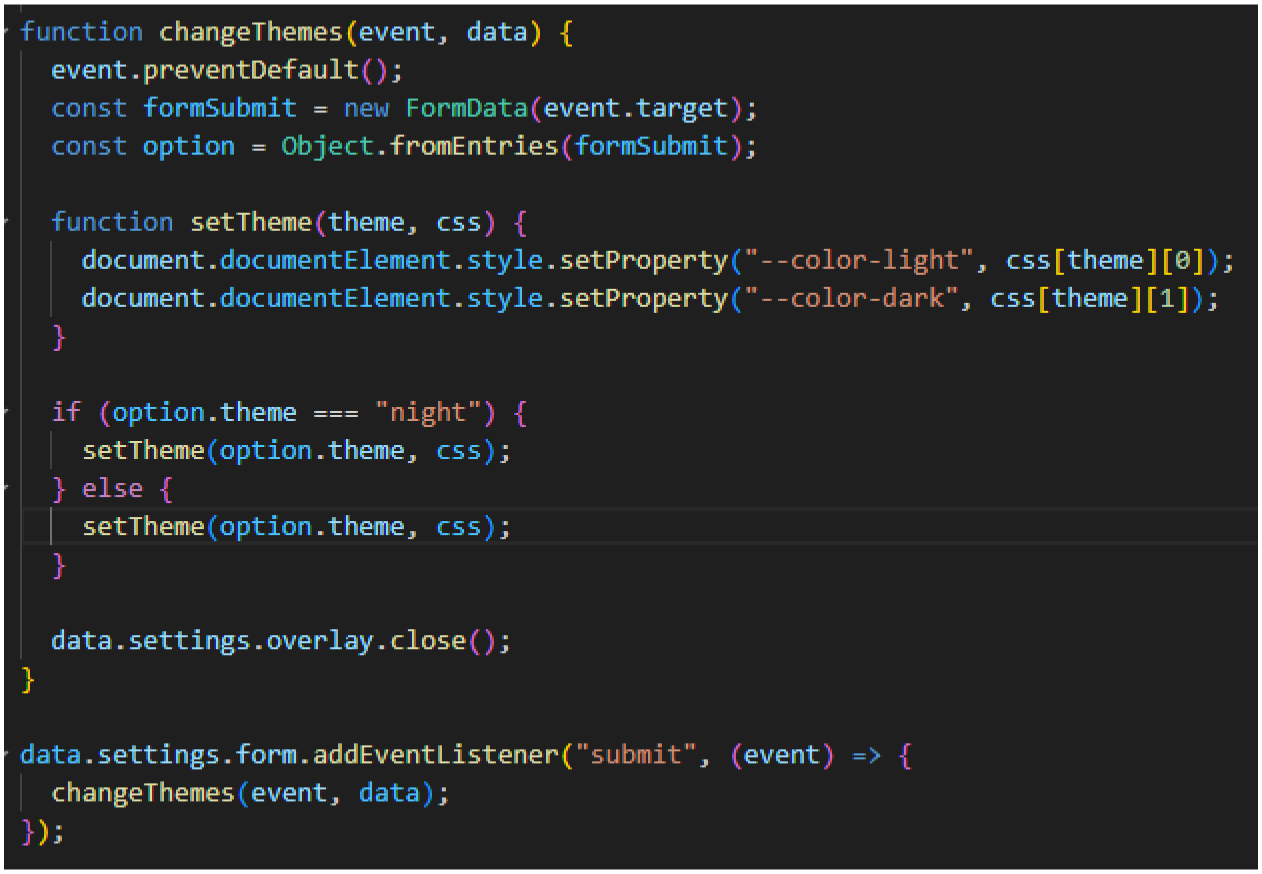
```
const genreList = Object.entries(genres);
for (let i = 0; i < genreList.length; i++) {
  const [id, name] = genreList[i];
  const genreOption = document.createElement("option");
  genreOption.value = id;
  genreOption.textContent = name;
  genresFragment.appendChild(genreOption);
}
data.search.genres.appendChild(genresFragment);
```



```
const authorArray = Object.entries(authors);
for (let i = 0; i < authorArray.length; i++) {
  const [id, name] = authorArray[i];
  const authorOption = document.createElement("option");
  authorOption.value = id;
  authorOption.textContent = name;
  authorsFragment.appendChild(authorOption);
}
data.search.authors.appendChild(authorsFragment);
```

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

- Duplicate Key Values – Using the SRP Principle by separating the responsibility of changing the theme from conditional logic. The code still uses the conditional check to determine the theme but now calls the “setTheme” Function to do so.



```
function changeThemes(event, data) {
  event.preventDefault();
  const formSubmit = new FormData(event.target);
  const option = Object.fromEntries(formSubmit);

  function setTheme(theme, css) {
    document.documentElement.style.setProperty("--color-light", css[theme][0]);
    document.documentElement.style.setProperty("--color-dark", css[theme][1]);
  }

  if (option.theme === "night") {
    setTheme(option.theme, css);
  } else {
    setTheme(option.theme, css);
  }

  data.settings.overlay.close();
}

data.settings.form.addEventListener("submit", (event) => {
  changeThemes(event, data);
});
```

- Duplicate Fragments - Using the LSP Principle we can take one function that could be created to generate a fragment with default content, & callbacks can be added where needed with different inputs to give required outputs.