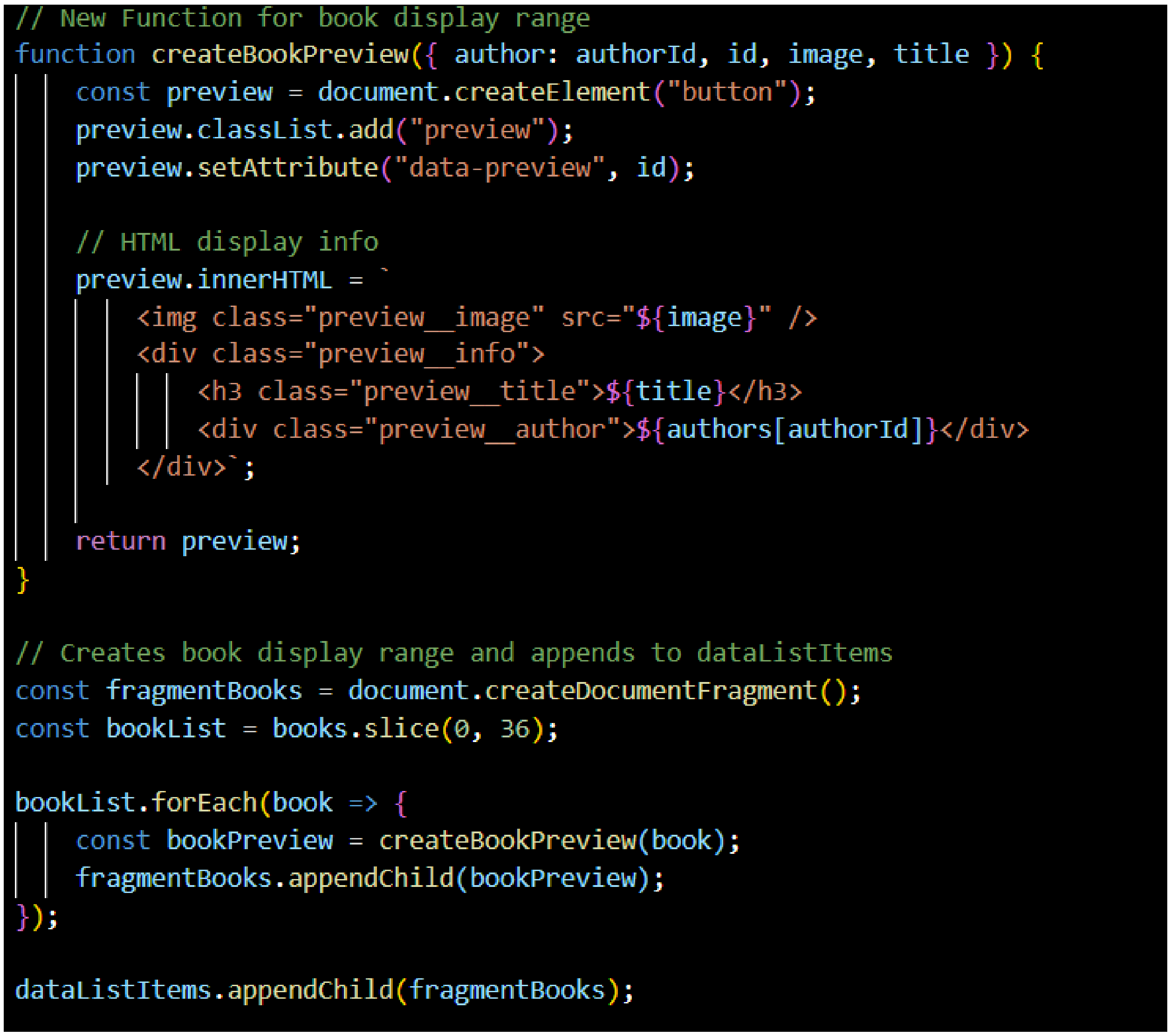
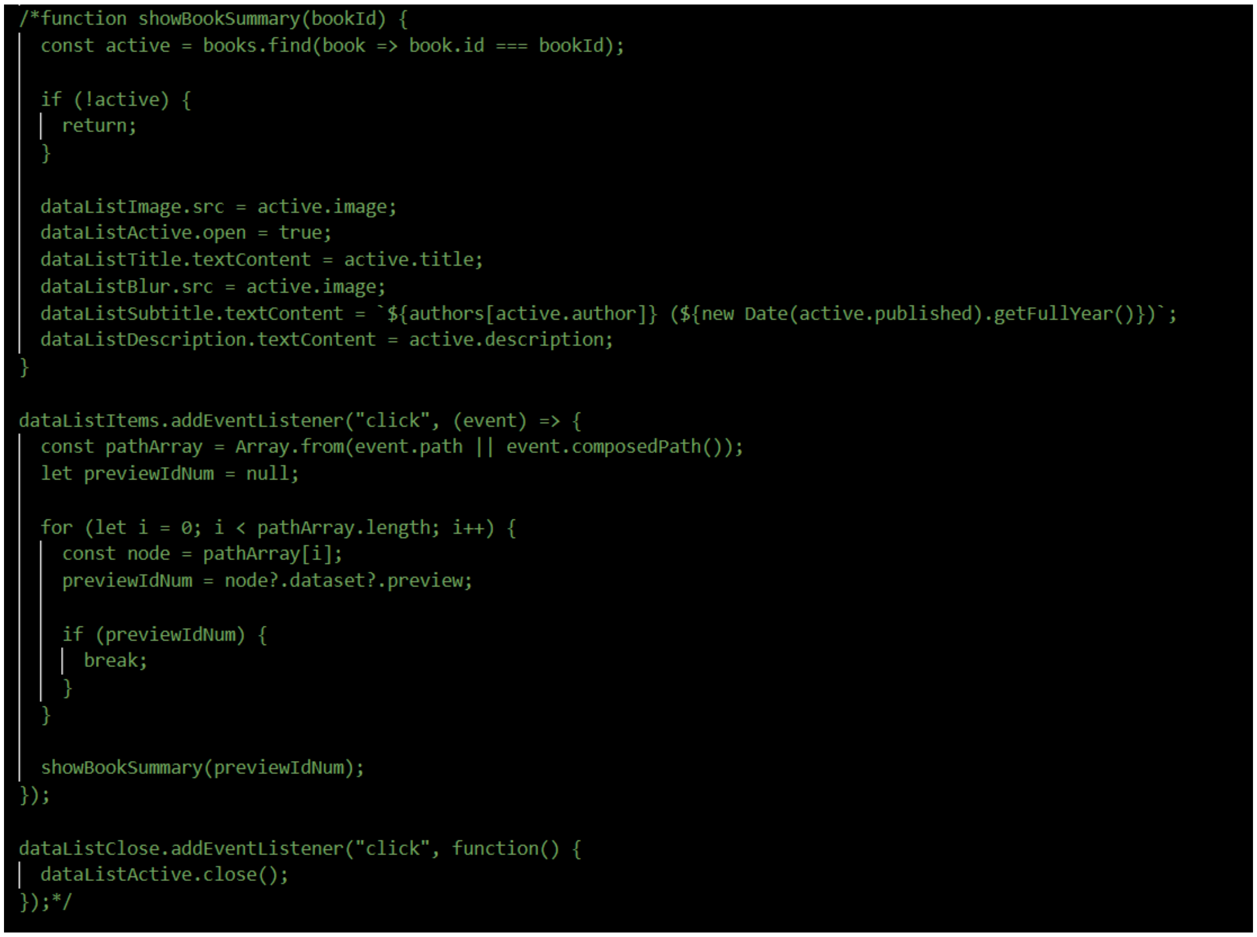
DWA_07.4 Knowledge Check_DWA7



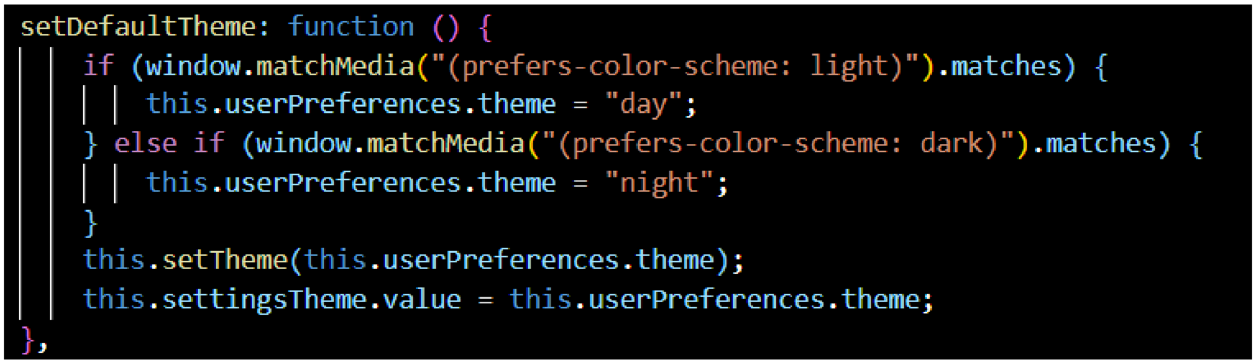
```javascript
// New Function for book display range
function createBookPreview({ author: authorId, id, image, title }) {
    const preview = document.createElement("button");
    preview.classList.add("preview");
    preview.setAttribute("data-preview", id);

    // HTML display info
    preview.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>`;

    return preview;
}

// Creates book display range and appends to dataListItems
const fragmentBooks = document.createDocumentFragment();
const bookList = books.slice(0, 36);

bookList.forEach(book => {
    const bookPreview = createBookPreview(book);
    fragmentBooks.appendChild(bookPreview);
});

dataListItems.appendChild(fragmentBooks);
```

This abstraction because it is singularly responsible for managing my Book display making it easier to manage and cleaning up my code.



```
/*function showBookSummary(bookId) {
  const active = books.find(book => book.id === bookId);

  if (!active) {
    return;
  }

  dataListImage.src = active.image;
  dataListActive.open = true;
  dataListTitle.textContent = active.title;
  dataListBlur.src = active.image;
  dataListSubtitle.textContent = `${authors[active.author]} (${new Date(active.published).getFullYear()})`;
  dataListDescription.textContent = active.description;
}

dataListItems.addEventListener("click", (event) => {
  const pathArray = Array.from(event.path || event.composedPath());
  let previewIdNum = null;

  for (let i = 0; i < pathArray.length; i++) {
    const node = pathArray[i];
    previewIdNum = node?.dataset?.preview;

    if (previewIdNum) {
      break;
    }
  }

  showBookSummary(previewIdNum);
});

dataListClose.addEventListener("click", function() {
  dataListActive.close();
});*/
```

This was the first initial funcion I did in my original submission for my capstone. It's short and simple to understand and handles only the function of showing the book summary.

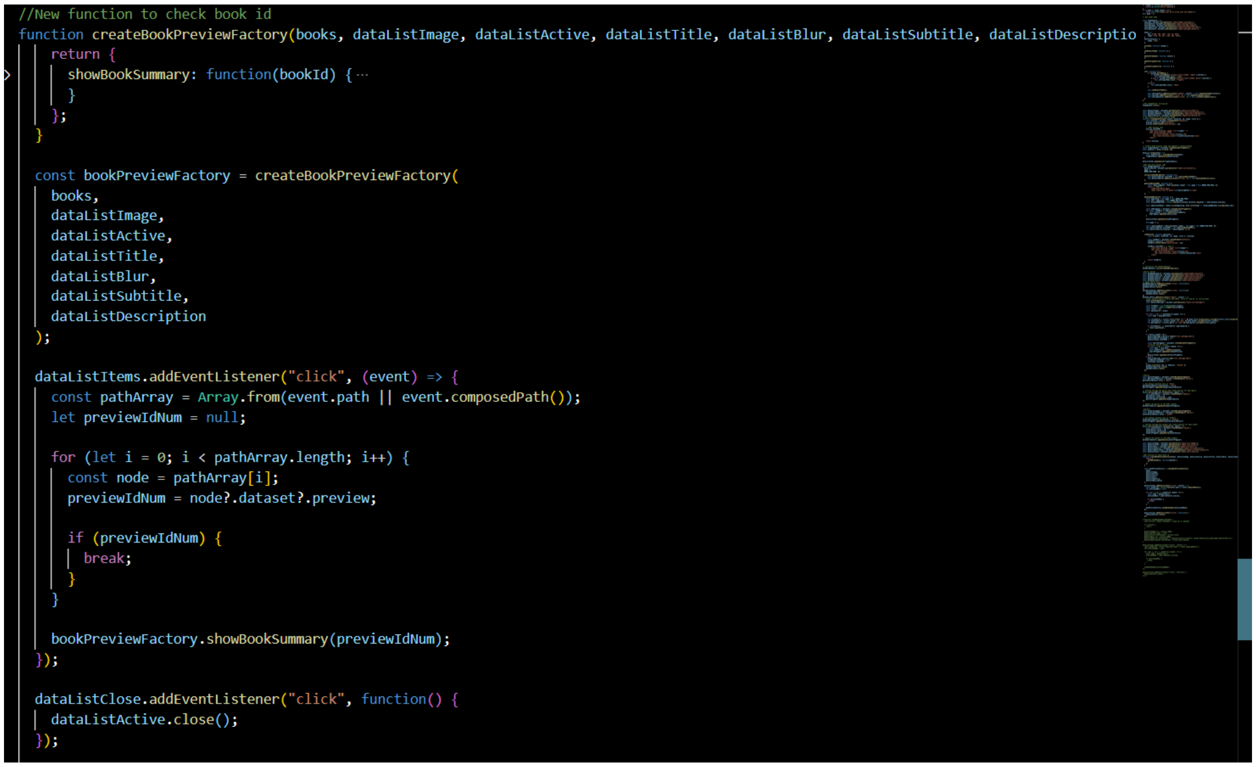


```
setDefaultTheme: function () {
    if (window.matchMedia("(prefers-color-scheme: light)").matches) {
        this.userPreferences.theme = "day";
    } else if (window.matchMedia("(prefers-color-scheme: dark)").matches) {
        this.userPreferences.theme = "night";
    }
    this.setTheme(this.userPreferences.theme);
    this.settingsTheme.value = this.userPreferences.theme;
},
```

Lastly, the function handling my dark/light theme. It handles a singular more selective task and isn't dependent on lower level modules.

---

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



```
//New function to check book id
function createBookPreviewFactory(books, dataListImage, dataListActive, dataListTitle, dataListBlur, dataListSubtitle, dataListDescriptio
    return {
      showBookSummary: function(bookId) {⋯
      }
    };
  }

  const bookPreviewFactory = createBookPreviewFactory(
    books,
    dataListImage,
    dataListActive,
    dataListTitle,
    dataListBlur,
    dataListSubtitle,
    dataListDescription
  );

  dataListItems.addEventListener("click", (event) => {
    const pathArray = Array.from(event.path || event.composedPath());
    let previewIdNum = null;

    for (let i = 0; i < pathArray.length; i++) {
      const node = pathArray[i];
      previewIdNum = node?.dataset?.preview;

      if (previewIdNum) {
        break;
      }
    }

    bookPreviewFactory.showBookSummary(previewIdNum);
  });

  dataListClose.addEventListener("click", function() {
    dataListActive.close();
  });
```

It does too much and is too dependent on other modules and has made my code harder to read instead of easier.

---

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

Single Responsibility Principle:
My abstraction handels too many things and can be further simplified
Open/Closed Principle:
Some of my better abstractions also have this problem and shouldn't be open to modification.

---

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