DWA_08 Discussion Questions

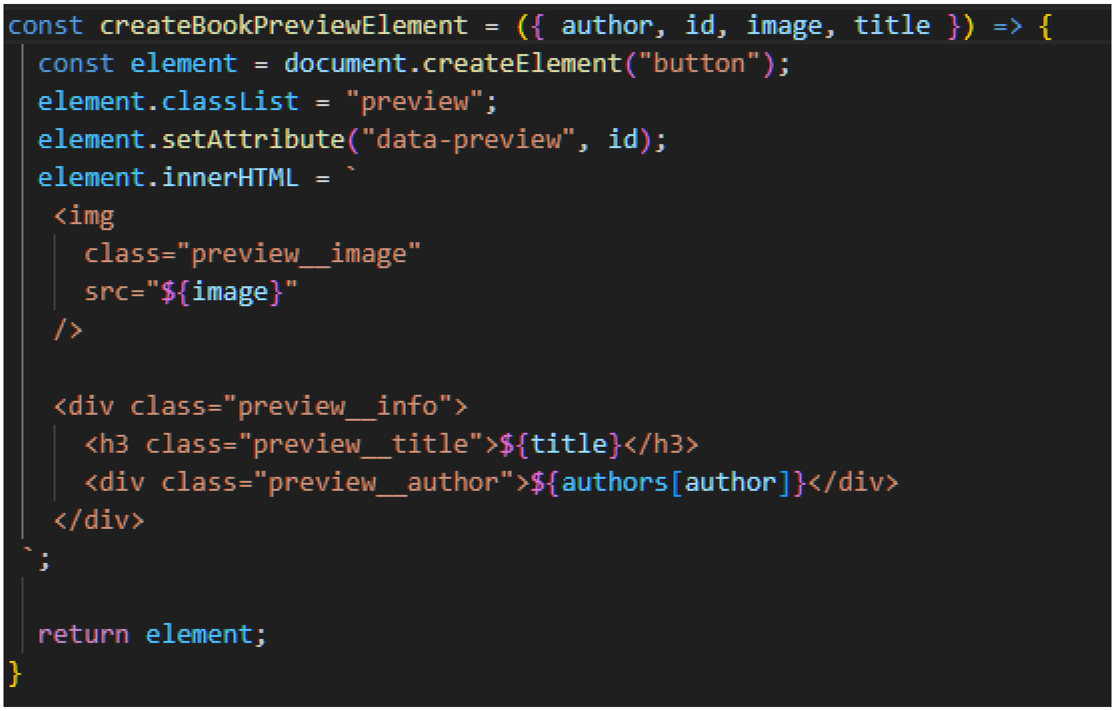
In this module you will continue with your “Book Connect” codebase, and further iterate on your abstractions. You will be required to create an encapsulated abstraction of the book preview by means of a single factory function. If you are up for it you can also encapsulate other aspects of the app into their own abstractions.

To prepare for your session with your coach, please answer the following questions. Then download this document as a PDF and include it in the repository with your code.

---

1. What parts of encapsulating your logic were easy?

Basically the same code as my previous updateDisplay() function.

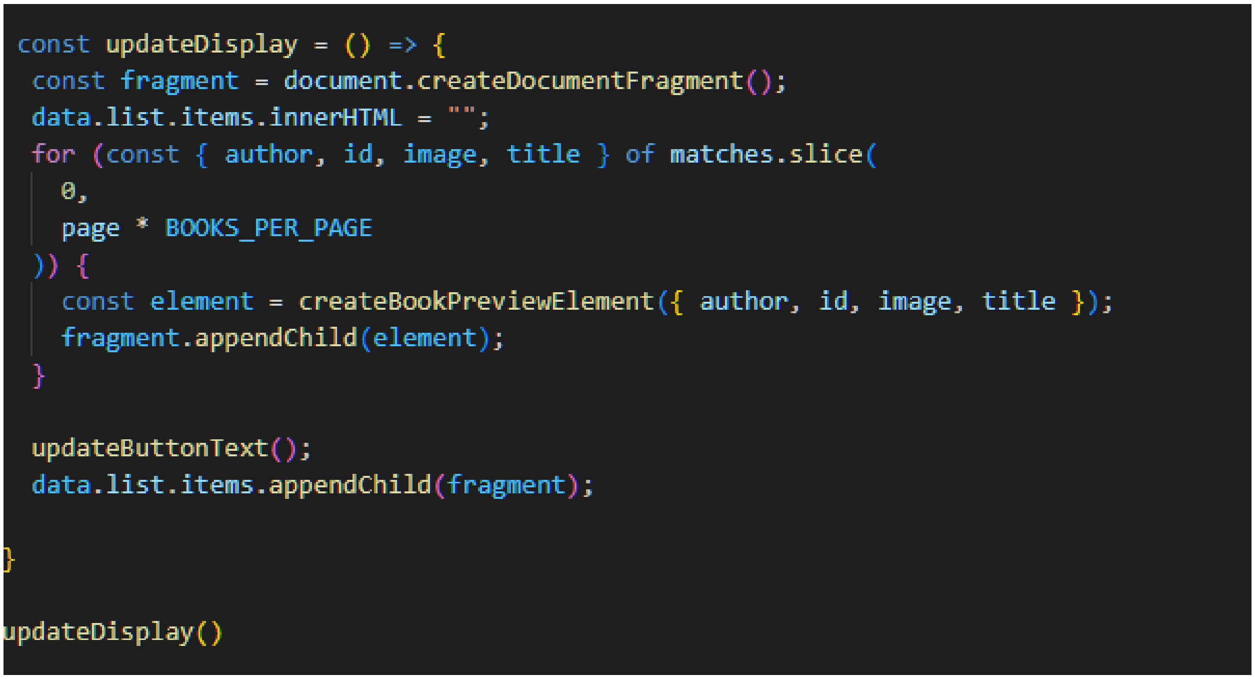


```
const createBookPreviewElement = ({ author, id, image, title }) => {
  const element = document.createElement("button");
  element.classList = "preview";
  element.setAttribute("data-preview", id);
  element.innerHTML = `
   <img
     class="preview__image"
     src="${image}"
   />

   <div class="preview__info">
     <h3 class="preview__title">${title}</h3>
     <div class="preview__author">${authors[author]}</div>
   </div>
 `;

  return element;
}
```

---

2. What parts of encapsulating your logic were hard?

While not hard per say, code needed to be moved around to work.



```
 const updateDisplay = () => {
  const fragment = document.createDocumentFragment();
  data.list.items.innerHTML = "";
  for (const { author, id, image, title } of matches.slice(
    0,
    page * BOOKS_PER_PAGE
  )) {
    const element = createBookPreviewElement({ author, id, image, title });
    fragment.appendChild(element);
  }

  updateButtonText();
  data.list.items.appendChild(fragment);

}

updateDisplay()
```

---

3. Is abstracting the book preview a good or bad idea? Why?

Yes, having a factory function that focuses on element creation and leaving DOM manipulation to separate logic helps me easily understand what each part of the code does.
