Diego Jeftha

ITW 19

Capstone Presentation

Used the ‘querySelector’ method of the ‘document’ object to select elements from the HTML document and assigned them to the corresponding values

```
const dataHeaderSearch = document.querySelector('[data-header-search]')
const dataHeaderSettings = document.querySelector('[data-header-settings]')
const dataListItems = document.querySelector('[data-list-items]')
const dataListMessage = document.querySelector('[data-list-message]')
const dataListButton = document.querySelector('[data-list-button]')
const dataListActive = document.querySelector('[data-list-active]')
const dataListBlur = document.querySelector('[data-list-blur]')
const dataListImage = document.querySelector('[data-list-image]')
const dataListTitle = document.querySelector('[data-list-title]')
const dataListSubtitle = document.querySelector('[data-list-subtitle]')
const dataListDescription = document.querySelector('[data-list-description]')
const dataListClose = document.querySelector('[data-list-close]')
const dataSearchOverlay = document.querySelector('[data-search-overlay]')
const dataSearchForm = document.querySelector('[data-search-form]')
const dataSearchTitle = document.querySelector('[data-search-title]')
const dataSearchGenres = document.querySelector('[data-search-genres]')
const dataSearchAuthors = document.querySelector('[data-search-authors]')
const dataSearchCancel = document.querySelector('[data-search-cancel]')
const dataSettingsOverlay = document.querySelector('[data-settings-overlay]')
const dataSettingsForm = document.querySelector('[data-settings-form]')
const dataSettingsTheme = document.querySelector('[data-settings-theme]')
const dataSettingsCancel = document.querySelector('[data-settings-cancel]')
```

This for loop creates a list of book previews and extracts the first 36 books from the books array

```
const fragment = document.createDocumentFragment()
const extracted = books.slice(0, 36)


for (let i = 0; i < extracted.length; i++) {
    const { author: authorId, id, image, title } = extracted[i]
    const element = document.createElement('button')          //creates a new button
    element.classList = 'preview'
    element.setAttribute('data-preview', id)
    element.innerHTML = /* html */ `
        <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)        // appends button element to fragment doc
}

dataListItems.appendChild(fragment)
```

Creates a dropdown list of book genres, creates a new document named ‘genreOfBook’ to store the option element

```
const genreOfBook = document.createDocumentFragment()
const optionOfBooks = document.createElement('option')
optionOfBooks.value = 'any'
optionOfBooks.textContent = 'All Genres'
genreOfBook.appendChild(optionOfBooks)
```

```
for (let [id, name] of Object.entries(genres)) {
    const optionsOfGenre = document.createElement('option')
    optionsOfGenre.value = id
    optionsOfGenre.textContent = name
    genreOfBook.appendChild(optionsOfGenre)
}
```

```
dataSearchGenres.appendChild(genreOfBook)
```

Loop loops over the genre object and each time it loops it creates a new option element and sets it’s value attribute to current id and textContent to current name and appends option element to ‘genreOfBook’

Appends ‘genreOfBook’ to dataSearchGenres

Show more button

```
dataListButton.addEventListener('click', () => {

    const start = page * BOOKS_PER_PAGE;
    const end = start + BOOKS_PER_PAGE;
    const newBook = books.slice(start, end)
    const newBookFragment = document.createDocumentFragment();

    for (let i = 0; i < newBook.length; i++) {
        const moreBooks = newBook[i];
        const showPreview = createPreview(moreBooks);
        newBookFragment.appendChild(showPreview);
    }
    dataListItems.appendChild(newBookFragment);

    const remaining = matches.length - page * BOOKS_PER_PAGE;
    dataListButton.innerHTML = /* html */ `
        <span>Show more</span>
        <span class="list__remaining"> (${remaining > 0 ? remaining : 0})</span
    `;
    dataListButton.disabled = remaining <= 0;
    page = page + 1
});
```

Calculates the start and end indices of the amount of books to be displayed on the current page and number of books per page

Loops over each element in the ‘newBook’ array using a for loop

Shows the remaining number of books, if the remaining number is less than or = to 0 the button is disabled

Increments page variable by 1

```
function createPreview(preview) {
    const { author: authorId, id, image, title } = preview;
    const moreBooks = document.createElement('button');
    moreBooks.classList = 'preview';
    moreBooks.setAttribute('data-preview', id);

    moreBooks.innerHTML = /* html */ `
        <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 moreBooks

}
```

Used destructuring to extract properties from preview object

Adds preview class to moreBooks button

Theme (Settings button)

```
dataSettingsForm.addEventListener("submit", (event) => {
    event.preventDefault();
    const formSubmit = new FormData(event.target)
    const option = Object.fromEntries(formSubmit)

    if (option.theme === 'night') {
        document.documentElement.style.setProperty('--color-light', css[option.theme][1])
        document.documentElement.style.setProperty('--color-dark', css[option.theme][0])
    } else if (option.theme === 'day') {
        document.documentElement.style.setProperty('--color-light', css[option.theme][1])
        document.documentElement.style.setProperty('--color-dark', css[option.theme][0])
    }
    dataSettingsOverlay.close()
})
```

When the ‘dataHeaderSettings’ button is clicked, function focuses on the input field and checks if the settings overlay is open or not

```
dataHeaderSettings.addEventListener('click', () => {
    dataSettingsTheme.focus();

    if (dataSettingsOverlay.open) {
        dataHeaderSettings.showModal()
    } else {
        dataSettingsOverlay.showModal();
    }
})

dataSettingsCancel.addEventListener('click', () => {
    dataSettingsOverlay.close();
})
```

When the cancel button is clicked it closes the overlay

When the ‘dataHeaderSearch’ button is clicked, function focuses on the input field and checks if the search overlay is open or not

```
dataHeaderSearch.addEventListener('click', () => {
    dataSearchTitle.focus();

    if (dataSearchOverlay.open) {
        dataHeaderSearch.showModal()
    } else {
        dataSearchOverlay.showModal();
    }
})

dataSearchCancel.addEventListener('click', () => {
    dataSearchOverlay.close();
})
```

When the cancel button is clicked it closes the overlay

Book filters

Declares an empty array `booksList` which will be used to store the search results

```
let booksList = [];

dataSearchForm.addEventListener('submit', (event) => {
    event.preventDefault();    // prevents default form submit behavior
    const formData = new FormData(event.target);
    const filters = Object.fromEntries(formData);
    const result = [];
```

```
for (const book of books) {
        const titleMatch = filters.title.trim() !== '' && book.title.toLowerCase().includes(filters.title.toLowerCase());  // use || instead of &&
        const authorMatch = filters.author !== 'any' && book.author.includes(filters.author)
        const genreMatch = filters.genre !== 'any' && book.genres.includes(filters.genre)

        if (titleMatch || authorMatch || genreMatch) {
            result.push(book);
        }
    }

    if (result.length > 0) {
        dataListMessage.classList.remove('list__message_show');
        dataListButton.disabled = true;
        dataListItems.innerHTML = '';

        const searchBooks = document.createDocumentFragment();

        for (let i = 0; i < result.length; i++) {
            const book = result[i];
            const bookPreview = createPreview(book);
            searchBooks.appendChild(bookPreview);
        }
        dataListItems.appendChild(searchBooks)
    } else {
        dataListMessage.classList.add('list__message_show');
        dataListButton.disabled = true;
        dataListItems.innerHTML = '';
    }
```

Used a for loop that loops over each book in the books array

Checks if 'titlematch" or 'authorMatch' or 'genreMatch' is true, if any are true the book is added to the result array using the push method

Creates an empty document fragment named 'searchBooks'

For loop loops over the result array and calls the function 'createPreview' with the argument of the current book then appends bookPreview to the searchBooks fragment. Appends searchBokks to dataListItems

Book Preview

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

    for (const node of pathArray) {
        if (active) {
            break;
        }
        const previewId = node?.dataset?.preview

        for (const singleBook of books) {
            if (singleBook.id === previewId) {
                active = singleBook
            }
        }
    }

    if (!active) {
        return;
    }
    dataListActive.open = true;
    dataListImage.src = active.image;
    dataListBlur.src = active.image;
    dataListTitle.textContent = active.title;

    dataListSubtitle.textContent = `${authors[active.author]} (${new Date(active.published).getFullYear()})`
    dataListDescription.textContent = active.description;
})
```

When a book is clicked on, it brings up a preview of the book

```
dataListClose.addEventListener('click', () => {
    dataListActive.close();
})
```

When the closed button is clicked it closes the book preview