Title of presentation: Debugging a Book Preview App

By Mothusi Mathuloe

Introduction

In this presentation, I will be discussing common bugs and issues that affected the functionality of JPreview Book App. We'll explore some examples of code snippets that have bugs and analyze how these issues can cause problems in the intended functionality of the code.

Outline

- Introduction
- User story 1: Previews of books
- User story 2: Author and genre dropdown options in the search form
- User story 3: Theme settings
- User story 4: Search functionality
- Additional functionality: Home button
- Conclusion

User story 1: Previews of books

1. **Overview of the original previews of books code:**

   1.1. This code is attempting to create a document fragment with previews of 36 books from an array of books called "books" and then appending that fragment to an element with the id "data-list-items".

2. **Bugs that I found:**

   2.1. The object destructuring syntax in the for loop is incorrect. The variables should be declared using let or const and the i variable should be initialized.

   2.2. There is no definition or reference to the createPreview function used in the loop, so it is likely that this code will throw an error.

3. **How the bugs affected the code**

   3.1. These bugs affect the intended functionality of the code by preventing it from correctly iterating over the first 36 items in the books array and creating previews for them. Additionally, the missing createPreview function may prevent the previews from being created correctly.

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

for ({ author, image, title, id }; extracted; i++) {
  const preview = createPreview({
    author,
    id,
    image,
    title
  })

  fragment.appendChild(preview)
}

data-list-items.appendChild(fragment)
```

Solution

```
43
44  const createPreview = ({ author, id, image, title }) => {
45      const preview = document.createElement('button');
46      preview.classList = 'preview';
47      preview.setAttribute('data-preview', id);
48
49      preview.innerHTML = /* html */ `
50        <img class="preview__image" src="${image}" alt="${title}">
51        <div class="preview__content">
52          <h2 class="preview__title">${title}</h2>
53          <h3 class="preview__author">${authors[author]}</h3>
54        </div>
55      `;
56
57      return preview;
58    };
59
60    let fragment = document.createDocumentFragment();
61    let extracted = matches.slice(range[0], range[1]);
62
63    for (const { author, image, title, id } of extracted) {
64      const preview = createPreview({
65        author,
66        id,
67        image,
68        title,
69      });
70
71      fragment.appendChild(preview);
72    }
73
74    items.append(fragment)
75
```

- I started off by importing all the objects in the data.js file so that the scripts.js file could java access to the data.
- I then defined the preview function and all the variables since they were not defined in the original code.
- I also destructured the object in the for loop by declaring the variables using const.
- I then removed the i afterthought or counter because it was not required in this solution since I used a ‘for.. Of’ loop.

User story 2: Author & Genre search options

1. **Overview of the original previews of books code:**

   1.1. The code aims to create two dropdown menus for filtering books by genre and author.

2. **Bugs that I found:**

   2.1. In the for loop for genres, the iteration variable is incorrectly named id, while it should be named entry. Also, the value and text variables used in the loop are not defined. Instead, they should be assigned the values of id and name respectively, from the current iteration entry using destructuring syntax.

   2.2. data-search-genres and data-search-authors are not defined in the code

   2.3. The same issues as in steps 1 to 3 occur in the for loop for authors.

```
genres = document.createDocumentFragment()
element = document.createElement('option')
element.value = 'any'
element = 'All Genres'
genres.appendChild(element)

for ([id, name]; Object.entries(genres); i++) {
    document.createElement('option')
    element.value = value
    element.innerText = text
    genres.appendChild(element)
}

data-search-genres.appendChild(genres)

authors = document.createDocumentFragment()
element = document.createElement('option')
element.value = 'any'
element.innerText = 'All Authors'
authors.appendChild(element)

for ([id, name];Object.entries(authors); id++) {
    document.createElement('option')
    element.value = value
    element = text
    authors.appendChild(element)
}

data-search-authors.appendChild(authors)
```

Solution

```
77
78  const genresFragment = document.createDocumentFragment()
79  let element = document.createElement('option')
80  element.value = 'any'
81  element.innerText = 'All Genres'
82  genresFragment.appendChild(element)
83
84  for (const [id, name] of Object.entries(genres)) {
85      let element = document.createElement('option')
86      element.value = id
87      element.innerText = name
88      genresFragment.appendChild(element)
89  }
90
91  searchGenre.appendChild(genresFragment)
92
93  const authorsFragment = document.createDocumentFragment()
94  let element2 = document.createElement('option')
95  element2.value = 'any'
96  element2.innerText = 'All Authors'
97  authorsFragment.appendChild(element2)
98
99  for (const [id, name] of Object.entries(authors)) {
100     let element2 = document.createElement('option')
101     element2.value = id
102     element2.innerText = name
103     authorsFragment.appendChild(element2)
104 }
105
106 searchAuthor.appendChild(authorsFragment)
107
108
```

- I defined all the variable that we not defined.
- I then fixed the for loops of both genres and authors.
- I also destructured the object in the for loop by declaring the variables using const and also removing the i afterthought or counter.
- I assigned both element.value and element2.value to id instead of value like they did in the original code.

User story 3: Theme settings

```
data - settings - theme.value === window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches ? 'night' : 'day'
v = window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches ? 'night' | 'day'

documentElement.style.setProperty('--color-dark', css[v].dark);
documentElement.style.setProperty('--color-light', css[v].light);
```

1. **Overview of the original previews of books code:**

   1.1. The code seems to be attempting to check if the user's system preference is set to a dark mode, and if so, set the theme of the webpage accordingly.

2. **Bugs that I found:**

   2.1. There is a syntax error on the first line of code. The variable name should not contain spaces.

   2.2. The second line of code is unclear and contains a syntax error. It seems to be attempting to set a variable v to either 'night' or 'day' based on whether the user's system preference is set to dark mode or not. However, the | operator is used instead of the : operator to set the value based on the condition

   2.3. The setProperty() method is used to set the --color-dark and --color-light CSS custom properties on the documentElement, but the values for these properties are obtained from an undefined css variable

Solution

- I defined all the variable that had to be defined using const.
- I changed the bitwise operator '|' in the v constant to the ':' operator to set the value based on the condition.
- The css variable deserves a special mention because without declaring it this function wouldn't have worked and this is one of the problems I spent most of my time trying to solve because I kept an missing it when I was looking for potential bugs in my code.

```
settingsTheme.value = window.matchMedia('(prefers-color-scheme: dark)').matches ? 'night' : 'day';
const v = window.matchMedia('(prefers-color-scheme: dark)').matches ? 'night' : 'day';

if (settingsTheme.value === 'night') {
  document.documentElement.style.setProperty('--color-dark', css['night'].dark);
  document.documentElement.style.setProperty('--color-light', css['day'].light);
}

if (v === 'night') {
  document.documentElement.style.setProperty('--color-dark', css['night'].dark);
  document.documentElement.style.setProperty('--color-light', css['day'].light);
}

settingsOverlay.addEventListener('submit', (event) => {
  event.preventDefault();
  const formData = new FormData(event.target);
  const result = Object.fromEntries(formData);
  document.documentElement.style.setProperty('--color-dark', css[result.theme].dark);
  document.documentElement.style.setProperty('--color-light', css[result.theme].light);
  settingsOverlay.open = false;
});
```

User story 4: Search functionality

```
data - search - form.click(filters) {
    preventDefault()
    const formData = new FormData(event.target)
    const filters = Object.fromEntries(formData)
    result = []

    for (book; booksList; i++) {
        titleMatch = filters.title.trim() = '' && book.title.toLowerCase().includes[filters.title.toLowerCase()]
        authorMatch = filters.author = 'any' || book.author === filters.author

        {
            genreMatch = filters.genre = 'any'
            for (genre; book.genres; i++) { if singleGenre = filters.genre { genreMatch === true } }
        }
    }

    if titleMatch && authorMatch && genreMatch => result.push(book)
}

if display.length < 1
    data - list - message.class.add('list__message_show')
else data - list - message.class.remove('list__message_show')


data - list - items.innerHTML = ''
const fragment = document.createDocumentFragment()
const extracted = source.slice(range[0], range[1])

for ({ author, image, title, id }; extracted; i++) {
    const { author: authorId, id, image, title } = props

    element = document.createElement('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)
}

data - list - items.appendChild(fragments)
initial === matches.length - [page * BOOKS_PER_PAGE]
remaining === hasRemaining ? initial : 0
data - list - button.disabled = initial > 0

data - list - button.innerHTML = /* html */ `
        <span>Show more</span>
        <span class="list__remaining"> (${remaining})</span>
    `


window.scrollTo({ top: 0, behavior: 'smooth' });
data - search - overlay.open = false
}
```

Overview of the original previews of books code:

1.1. This code allows the user to filter the books/ search for books based on the title, and also author or genre by selecting an option from the dropdown options.

Bugs that I found:

2.1. There are several syntax errors in the code, including missing parentheses, curly braces, and semicolons. These errors would prevent the code from running properly.

2.2. Several variables are used without being defined or assigned a value, including data-search-form, booksList, singleGenre, display, source, range, props, authors, matches, page, BOOKS_PER_PAGE, and hasRemaining.

2.3. There are several logical errors in the code, including incorrect use of comparison and assignment operators, and incorrect conditional statements.

2.4. The code appears to define a function to handle a form submission event (data-search-form.click(filters)), but it does not attach an event listener to the form to call the function

2.5. The code attempts to manipulate DOM elements by setting their innerHTML property and using class and disabled properties, but it does not define these elements or select them from the DOM.

Solution

- I fixed all the syntax errors in the code, this included missing parentheses, curly braces, and semicolons.
- I also fixed the for loop by initializing the i iterator and also changing the condition to i < matches.length.
- I also removed the second for loop by using the ‘for…of’ loop in order to iterate over the result array instead of the extracted object like it was in the original code.
- I also fixed all the logical errors in the code by using the comarison “== / ===” where they are supposed to and using the assignment operator ‘=’ where it is supposed to be used.

```
searchForm.addEventListener('submit', (event) => {
  event.preventDefault();

  const formData = new FormData(searchForm);
  const filters = Object.fromEntries(formData);
  const result = [];

  for (let i = 0; i < matches.length; i++) {
    const book = matches[i];

    const titleMatch = filters.title.trim() === '' || book.title.toLowerCase().includes(filters.title.toLowerCase());
    const authorMatch = filters.author === 'any' || book.author === filters.author;
    const genreMatch = filters.genre === 'any' || book.genres.includes(filters.genre);

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

  const dataListMessage = document.querySelector('[data-list-message]');

  if (result.length < 1) {
    dataListMessage.classList.add('list__message_show');
    items.innerHTML = '';
  } else {
    dataListMessage.classList.remove('list__message_show');
    items.innerHTML = '';

    const fragment = document.createDocumentFragment();

    for (const book of result) {
      const { author, image, title, id } = book;

      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>
      `;

      fragment.appendChild(element);
    }

    items.appendChild(fragment);
  }

  searchOverlay.open = false;
  listBtn.disabled = true

});
```

Additional functionality: Home button

```
const div = document.querySelector('.header__logo');

  div.addEventListener('click', () => {
    // searchForm.reset(); // clear the search filters
    items.innerHTML = ''; // reset the search results to the default list
    listBtn.disabled = false
    for (const { author, image, title, id } of extracted) {
      //const { author: authorId, id, image, title } = props

      const element = document.createElement('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[author]}</div>
            </div>
          `
          fragment.appendChild(element)

      }

      items.appendChild(fragment)

  });
```

Problem

- After the user has used the search bar, and the books have been filtered according to their specification, if the user wants to go back to the list containing all the books it was no possible.

Solution

So to solve this problem I made the logo a button so that whenever the user wants to get the original book list they can simply press it.

Conclusion

Thank you for your time!