Introduction

- Title of presentation: Building a Book Preview App

  Brief explanation of what the presentation is about

User story 1: View Book Previews

```
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)
```

Original code

The code below was supposed to create a document fragment, extract the first 36 books from the books array, loop through each book and create a preview element for each book using the createPreview function, and finally append all the previews to the data-list-items element.

View Book Previews continued ...

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

    preview.innerHTML = /* html */ `
      <img class="preview__image" src="${image}" alt="${title}">
      <div class="preview__content">
        <h2 class="preview__title">${title}</h2>
        <h3 class="preview__author">${author}</h3>
      </div>
    `;

    return preview;
};

let fragment = document.createDocumentFragment();
let extracted = matches.slice(range[0], range[1]);

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

    fragment.appendChild(preview);
}
```

My code

User story 5: Text Phrase Search

- Description: As a user, I want to find books based on specific text phrases so that I don’t need to remember the entire title of a book.
- Image: An example of a search bar with the option to search by text phrase

User story 6: Author & Genres Search dropdown options

```
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)
```

- Description: As a user, I want to filter books by author so that I can find books to read by authors that I enjoy.

User story 7: 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
}
```

Original code

Search functionality continued

```
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

});
```

My code

User story 8: Dark and Light Modes

```
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);
data-list-button = "Show more (books.length - BOOKS_PER_PAGE)"

data-list-button.disabled = !(matches.length - [page * BOOKS_PER_PAGE] > 0)

data-list-button.innerHTML = /* html */ [
    '<span>Show more</span>',
    '<span class="list__remaining"> (${matches.length - [page * BOOKS_PER_PAGE] > 0 ? matches.length - [page * BOOKS_PER_PAGE] : 0})</span>',
]

data-search-cancel.click() { data-search-overlay.open === false }
data-settings-cancel.click() { querySelect(data-settings-overlay).open === false }
data-settings-form.submit() { actions.settings.submit }
data-list-close.click() { data-list-active.open === false }
```

- Description: As a user, I want to toggle between dark and light modes so that I can use the app comfortably at night.

Dark & light themes continued ...

```
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;
});
```

Conclusion

- Summary of the user stories presented
- The benefits of building an app that meets these user needs
- Call-to-action: Develop an app that meets these user stories