



# Book-connect IWA 19

By: Julian Quenet



# What to expect during this slideshow:

- 1: Going over code snippets
- 2: Miscellaneous changes to code
- 3: Challenges faced
- 4: Overall thoughts
- 5: Q&A if time provides

# Jumping right into it the code snippets:

File: domData.js

For retrieving and manipulating data from DOM and CSSOM

index.html # styles.css M JS data.js M JS scripts.js M JS domData.js M X

```
JS domData.js > ...
1 | import { genresObj, authors} from "../data.js";
2 |
3 | /**
4 |  * Object containing all query selectors
5 |  */
6 | export const selectors = {
7 |   list: document.querySelector("[data-list-items]"),
8 |   message: document.querySelector("[data-list-message]"),
9 |   loadMore: document.querySelector("[data-list-button]"),
10 |   previewOverlay: {
11 |     overlay: document.querySelector("[data-list-active]"),
12 |     overlayBtn: document.querySelector("[data-list-close]"),
13 |     overlayBlur: document.querySelector("[data-list-blur]"),
14 |     overlayImage: document.querySelector("[data-list-image]"),
15 |     titleOverlay: document.querySelector("[data-list-title]"),
16 |     dataOverlay: document.querySelector("[data-list-subtitle]"),
17 |     infoOverlay: document.querySelector("[data-list-description]"),
18 |   },
19 |   theme: {
20 |     themeBtn: document.querySelector("[data-header-settings]"),
21 |     themeOverlay: document.querySelector("[data-settings-overlay]"),
22 |     themeCancelBtn: document.querySelector("[data-settings-cancel]"),
23 |     themeForm: document.querySelector("[data-settings-form]"),
24 |     themeSelect: document.querySelector("[data-settings-theme]"),
25 |   },
26 |   search: {
27 |     searchBtn: document.querySelector("[data-header-search]"),
28 |     searchOverlay: document.querySelector("[data-search-overlay]"),
29 |     searchCancelBtn: document.querySelector("[data-search-cancel]"),
30 |     searchForm: document.querySelector("[data-search-form]"),
31 |   },
32 |   genresSelect: document.querySelector("[data-search-genres]"),
33 |   authorSelect: document.querySelector("[data-search-authors]"),
34 |   title: document.querySelector("[data-search-title]"),
35 | };
```

File Edit Selection View Go Run Terminal Help

domData.js - book-connect - Visual Studio Code

< index.html

# styles.css M

JS data.js M

JS scripts.js M

JS domData.js M X

JS domData.js > ...

35 };

36

37 export const css = {

38 day: {

39 dark: "10, 10, 20",

40 light: "255, 255, 255",

41 },

42 night: {

43 dark: "255, 255, 255",

44 light: "10, 10, 20",

45 },

46 };

47

48 document.querySelector(".overlay\_\_button").style.outline = 0; // Fixing the outline bug with the overlay close button

49

## File: domData.js

```
49
50
51
52 /**
53  * Function made to insert values from the inputted objects to the newly created 'option' element, a fragment is created
54  * as well and the new options elements with their values are appended to the fragment, the function then returns the
55  * fragment which can then be appended to the required parent element, the function also takes in a string argument
56  * which is primarily used to create a default option value i.e "All". To get the the values from the objects the function
57  * runs a loop through the object getting both the key values and the properties of those key values.
58  * @param {string} text
59  * @param {object} object
60  * @returns appended element
61  */
62 const optionsCreate = (text, object) => {
63   const fragment = document.createDocumentFragment();
64   const allOption = document.createElement("option");
65   allOption.value = "All";
66   allOption.innerText = text;
67   fragment.appendChild(allOption);
68
69   for (const [keyValue, property] of Object.entries(object)) {
70     const option = document.createElement("option");
71     option.value = keyValue;
72     option.innerText = property;
73     fragment.appendChild(option);
74   }
75
76   return fragment;
77 };
78
79 selectors.genresSelect.appendChild(optionsCreate("All genres", genresObj));
80 selectors.authorSelect.appendChild(optionsCreate("All authors", authors));
81
82 //Set the colors of the preview overlay text to correspond with the theme change
83 selectors.previewOverlay.titleOverlay.style.color = `rgba(var(--color-dark))`
84 selectors.previewOverlay.dataOverlay.style.color = `rgba(var(--color-dark))`
85 selectors.previewOverlay.infoOverlay.style.color = `rgba(var(--color-dark))`
```


```
index.html 2  # styles.css  JS data.js  JS scripts.js X
JS scripts.js > ...
1  import { BOOKS_PER_PAGE, authors, books } from "../data.js";
2  import { selectors, css } from "../domData.js";
3
4
5
6  /** Function made to create the innerHTML for the created element(booksElement) in the function and insert the values inputted
7   * to the areas required in order to display correctly in the html/DOM, this function will also take
8   * in the inputted values to add unique attributes to the created element i.e. class name and or
9   * dataset. The function returns the booksElement which can be later appended to the chosen parent element.
10  *
11  * @param {object} book
12  * @param {number} index
13  * @returns newly created element
14  */
15  const innerHTML = (book, index) => {
16    const booksElement = document.createElement("div");
17    booksElement.dataset.index = `${index}`; // Retrieving the index to make it easier to fetch data for future use
18    booksElement.className = "preview";
19    booksElement.id = book.id;
20    booksElement.innerHTML = `<img src = ${
21      book.image
22    } class = 'preview_image' alt="${book.title} book image"></img>
23    <div class="preview_info">
24      <h3 class="preview_title">${book.title}</h3>
25      <div class="preview_author">${authors[book.author]}</div>
26    </div>`;
27    return booksElement;
28  };
29
30  // Initial loading of the first 36 books
31  for (let i = 0; i < BOOKS_PER_PAGE; i++) {
32    selectors.list.appendChild(innerHTML(books[i], i));
33  }
34
35  // Changing the text content of the "Show more" button
36  selectors.loadMore.innerHTML = `<span>Show more</span>
37  <span class = "list_remaining">${(books.length - BOOKS_PER_PAGE)}</span>`;
```

## File: scripts.js

```
index.html 2  JS scripts.js M X  # styles.css  JS data.js

JS scripts.js > openOverlayHandler
40 let newlyLoaded = 0;
41 //-----All eventHandlers below-----
42
43 const moreBooksHandler = (e) => { // loads more books when the loadMore button is clicked
44   e.stopPropagation();
45   newlyLoaded += BOOKS_PER_PAGE; // Once clicked the newlyLoaded is incremented by BOOKS_PER_PAGE to keep track of books added
46   let booksLeft = books.length - BOOKS_PER_PAGE - newlyLoaded;
47   let btnText = booksLeft > 0 ? booksLeft : 0;
48   selectors.loadMore.querySelector(".list__remaining").textContent = `${btnText}`;
49
50   let booksLoaded = BOOKS_PER_PAGE + newlyLoaded;
51   //With concern to time complexity this event handler will only ever loop through and append a max of 36 items
52   for (let i = newlyLoaded; i < booksLoaded; i++) {
53     if (i === books.length) {
54       selectors.loadMore.disabled = true;
55       break;
56     } else {
57       selectors.list.appendChild(innerHTML(books[i], i));
58     }
59   }
60 };
61
62 // Opens the preview overlay when a preview is clicked and display all the same information as the preview and more
63 const openOverlayHandler = (e) => {
64   const overlay = selectors.previewOverlay.overlay;
65   const bookPreview = e.target.closest(".preview");
66   const index = bookPreview.dataset.index; // The index is used to retrieve the corresponding data
67
68   selectors.previewOverlay.overlayBlur.src = books[index].image;
69   selectors.previewOverlay.overlayImage.src = books[index].image;
70   selectors.previewOverlay.titleOverlay.textContent = books[index].title;
71   let dateOverlay = new Date(books[index].published).getFullYear();
72   selectors.previewOverlay.dataOverlay.textContent = `${
73     authors[books[index].author]
74   } (${dateOverlay})`;
75   selectors.previewOverlay.infoOverlay.textContent = books[index].description;
76
77   overlay.show();
78 };
79
```

## File: scripts.js



```
File Edit Selection View Go Run Terminal Help scripts.js - book-connect - Visual Studio Code
index.html styles.css M JS data.js M JS scripts.js M X
JS scripts.js > searchSubmitHandler
80
81 // Opens the theme settings and sets it's values
82 const themeToggleHandler = (e) => {
83 // Checks to see if backgroundColor matches that of the set 'night' color scheme
84 const darkMode = getComputedStyle(document.body).backgroundColor === `rgb(${css.night.light})`;
85 selectors.theme.themeSelect.value = darkMode ? "night" : "day";
86
87 const overlay = selectors.theme.themeOverlay;
88 const closeBtn = selectors.theme.themeCancelBtn;
89 overlay.show();
90 if (e.target === closeBtn) {
91 overlay.close();
92 }
93 };
94
95 // Changes the color scheme when the form values have been saved/submitted
96 const themeSubmitHandler = (e) => {
97 e.preventDefault();
98
99 const overlay = selectors.theme.themeOverlay;
100 const formData = new FormData(e.target);
101 const themeChoice = Object.fromEntries(formData);
102 const theme = themeChoice.theme;
103 document.documentElement.style.setProperty("--color-dark", css[theme].dark);
104 document.documentElement.style.setProperty("--color-light", css[theme].light);
105 overlay.close();
106 };
107
108
```



## File: scripts.js

```
index.html # styles.css M JS data.js M JS scripts.js M X
JS scripts.js > ...
107 |
108 | let formValues; // Will only be truthy when the searchForm is submitted
109 |
110 | // Opens/closes the filter form
111 | const searchToggleHandler = (e) => {
112 |   const overlay = selectors.search.searchOverlay;
113 |   const closeBtn = selectors.search.searchCancelBtn;
114 |   overlay.show();
115 |   if (formValues) { // The values are based on what was entered into the form
116 |     selectors.genresSelect.value = formValues.genre;
117 |     selectors.authorSelect.value = formValues.author;
118 |     selectors.title.value = formValues.title;
119 |   }
120 |   if (e.target === closeBtn) {
121 |     overlay.close();
122 |     selectors.search.searchForm.reset()
123 |   }
124 | };
125 |
126 | let filteredBooks; // Will only be truthy when searchForm is submitted
127 | let filteredLoad; // Will only receive a value when the searchForm is submitted
128 |
```

## File: scripts.js

```
JS scripts.js > [e] searchSubmitHandler
128
129 const searchSubmitHandler = (e) => {
130   e.preventDefault();
131   const overlay = selectors.search.searchOverlay;
132   const formData = new FormData(e.target);
133   const filters = Object.fromEntries(formData);
134   const result = [];
135   books.forEach((book, index) => {
136     const { title, author, genres } = book;
137     const categories = [...genres]; // Spread operator to make sifting through the data easier instead of using a for of loop
138
139     const genreMatch = categories.includes(filters.genre) || filters.genre === "All";
140     const authorMatch = author === filters.author || filters.author === "All";
141     const titleMatch = title.toLowerCase().includes(filters.title.toLowerCase()) || filters.title === "";
142     // Only if all three are true will the data get pushed to the array
143     if (authorMatch && genreMatch && titleMatch) {
144       result.push([book, index]);
145     }
146   });
147   //-----Retrieving and manipulating data above this line-----
148   //-----Below this line: Conditionals and actions-----
149   const previews = selectors.list.querySelectorAll(".preview");
150   for (const book of previews) {
151     book.remove(); // Upon submission all previous books are removed
152   }
153
154   if (result.length === 0) { // If no matches are found the needed message pops up and loadMore button is disabled
155     selectors.message.classList.add("list__message_show");
156     selectors.loadMore.disabled = true;
157     selectors.loadMore.querySelector(".list__remaining").textContent = ` (0) `;
158   } else {
159     selectors.message.classList.remove("list__message_show");
160     selectors.loadMore.disabled = false;
161   }
162 }
```

## File: scripts.js

```
index.html 2 JS scripts.js X # styles.css JS data.js
JS scripts.js > ...
149 const previews = selectors.list.querySelectorAll(".preview");
150 for (const book of previews) {
151   book.remove(); // Upon submission all previous books are removed
152 }
153
154 if (result.length === 0) { // If no matches are found the needed message pops up and loadMore button is disabled
155   selectors.message.classList.add("list_message_show");
156   selectors.loadMore.disabled = true;
157   selectors.loadMore.querySelector(".list_remaining").textContent = `(0)`;
158 } else {
159   selectors.message.classList.remove("list_message_show");
160   selectors.loadMore.disabled = false;
161 }
162
163 if (result.length < BOOKS_PER_PAGE) { // Loads and appends books and disables the button
164   for (let i = 0; i < result.length; i++) {
165     let book = result[i][0];
166     let index = result[i][1];
167     selectors.list.appendChild(innerHTML(book, index));
168     selectors.loadMore.disabled = true;
169     selectors.loadMore.querySelector(".list_remaining").textContent = `(0)`;
170   }
171 } else {
172   // If there are more books than 36, then 36 are loaded, the rest are loaded with a "new" eventListener
173   for (let i = 0; i < BOOKS_PER_PAGE; i++) {
174     let book = result[i][0];
175     let index = result[i][1];
176     selectors.list.appendChild(innerHTML(book, index));
177     selectors.loadMore.querySelector(".list_remaining").textContent = `(${result.length - BOOKS_PER_PAGE})`;
178     selectors.loadMore.removeEventListener("click", moreBooksHandler); // Old eventListener is removed
179     filteredBooks = result;
180   }
181 }
182
183 overlay.close();
184 window.scrollTo({top: 0, behavior: "smooth"});
185 filteredLoad = 0; // Is given a value of zero each time it's loaded
186 formValues = filters; // formValues receives the same data as the filters variable i.e gets used in the searchToggleHandler
187 };
```

## File: scripts.js

```
index.html 2 JS scripts.js X # styles.css JS data.js
JS scripts.js > openOverlayHandler
177     filteredBooks = result;
180   }
181 }
182
183 overlay.close();
184 window.scrollTo({top: 0,behavior: "smooth"});
185 filteredLoad = 0; // Is given a value of zero each time it's loaded
186 formValues = filters; // formValues receives the same data as the filters variable i.e gets used in the searchToggleHandler
187 };
188
189
190 //Same concept as the moreBooksHandler but only runs if filteredBooks is truthy
191 const filterMoreHandler = (e) => {
192   if (!filteredBooks) {
193     return;
194   }
195   filteredLoad += BOOKS_PER_PAGE;
196   let booksLeft = filteredBooks.length - BOOKS_PER_PAGE - filteredLoad;
197   let btnText = booksLeft > 0 ? booksLeft : 0;
198   selectors.loadMore.querySelector(
199     ".list__remaining"
200   ).textContent = `${btnText}`;
201
202   let booksLoaded = BOOKS_PER_PAGE + filteredLoad;
203   for (let i = filteredLoad; i < booksLoaded; i++) {
204     if (i === filteredBooks.length) {
205       selectors.loadMore.disabled = true;
206       break;
207     } else {
208       let book = filteredBooks[i][0];
209       let index = filteredBooks[i][1];
210       selectors.list.appendChild(innerHTML(book, index));
211     }
212   }
213 };
214
215
```

## File: scripts.js

index.html

# styles.css M

JS data.js M

JS scripts.js M X

JS scripts.js > searchSubmitHandler

215

216

217

218 selectors.loadMore.addEventListener("click", moreBooksHandler);

219 selectors.loadMore.addEventListener("click", filterMoreHandler);

220 selectors.list.addEventListener("click", openOverlayHandler);

221 selectors.previewOverlay.overlayBtn.addEventListener("click", () => {

222 | selectors.previewOverlay.overlay.close();

223 | });

224

225 selectors.theme.themeBtn.addEventListener("click", themeToggleHandler);

226 selectors.theme.themeCancelBtn.addEventListener("click", themeToggleHandler);

227 selectors.theme.themeForm.addEventListener("submit", themeSubmitHandler);

228 selectors.search.searchBtn.addEventListener("click", searchToggleHandler);

229 selectors.search.searchCancelBtn.addEventListener("click", searchToggleHandler);

230 selectors.search.searchForm.addEventListener("submit", searchSubmitHandler);

231

# Miscellaneous changes:

```

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />

<link
  href="https://fonts.googleapis.com/css2?family=Roboto:wght@500;700&display=swap"
  rel="stylesheet"
/>

<script src="./scripts.js" defer type="module"></script>
<script src="./domData.js" defer type="module"></script>

```

```

91   "37018341-31f4-4ffa-8755-a49979c218dd": "Suzanne Collins"
92 }
93
94 // Changed the name as to avoid confusion with the genres key in the "book" object
95 export const genresObj = {
96   "a4f80b3e-3e96-4266-b729-e09b71793182": "Economics",
97   "6dd5bb6e-0172-4d6e-aa18-26f00954dd7a": "Non-fiction",
98   "5439a895-20a8-421a-981d-43f99b521cb5": "Business",
99   "e8ebb4be-b86b-49db-985c-2be996e4d5b8": "Science",
100  "90dd9a0f-9cce-48b7-b431-bde529a627bf": "Psychology",
101  "2afe9af7-23b7-4476-992e-609102106df5": "Sociology",
102  "39ca8a42-15aa-4774-ad4a-eda304b6ad56": "Audiobook",
103  "8c92b5be-e0ef-4d01-ad97-b93dfd404e08": "Self-help",
104  "45db37a5-83ea-4e2c-976d-dd962967bcf1": "Politics",
105  "746b98ed-b0b5-4bf3-bc1c-af8657601c7b": "Memoir",

```



## Challenges and obstacles I faced:

- 1: Time management and execution
- 2: The 'nitty gritty' of objects and arrays i.e. cans and cannots
- 3: Getting the code to do what I want
- 4: Overall structure of code



## Overall thoughts on project:

Was definitely intimidating at first but after I finished IWA18 it seemed a little easier, I took things step for step, wrote down everything from the brief and ticked them off. Overall I enjoyed it, though not confident about the overall structure of my code but it works and there aren't any bugs that I could find.

Q&A if there is time remaining