JavaScript DOM: Library Management System

Task Overview

Build a complete library management system that combines all JavaScript concepts we've learned with DOM manipulation. This task will test your understanding of variables, data types, operators, conditionals, arrays, loops, functions, objects, and DOM interaction.

Requirements

1. Data Structure

- Use the provided `library` array to store all books
- Each book object has these properties:
  - `id` (unique number)
  - `title` (string)
  - `author` (string)
  - `year` (number)
  - `available` (boolean)
  - `borrower` (string, empty if available)
  - `dueDate` (date string, empty if available)

2. Core Functions to Implement

Replace the TODO comments in the JavaScript code with working implementations:

`addBook(title, author, year)`

- Validate that all fields are filled
- Create a new book object with `generateId()`
- Add to library array
- Call `displayAllBooks()` to update the UI
- Return success/error message

`removeBook(bookId)`

- Validate the book ID (must be a number)
- Find the book in the library
- Only remove if the book is available
- Update the DOM display
- Return success/error message

`searchBooks(query)`

- Search books by title OR author (case-insensitive)
- Return array of matching books
- Use `includes()` method for partial matches

`borrowBook(bookId, borrowerName)`

- Validate inputs (ID must be number, name not empty)
- Find book and check if available
- Set `available: false`, `borrower`, and `dueDate` (14 days from today)
- Update DOM display
- Return success/error message

`returnBook(bookId)`

- Validate book ID
- Find book and check if it's borrowed
- Calculate overdue fee using `calculateOverdueFee()`
- Reset book status (`available: true`, clear borrower and dueDate)
- Update DOM display
- Return message with fee information

`displayAllBooks()`

- Get the `books-display` element
- Clear existing content (`innerHTML = ''`)
- Loop through library and create DOM elements
- Use `createBookElement()` for each book
- Append elements to the container

`calculateOverdueFee(dueDate)`

- Parse the due date string
- Calculate days between today and due date
- Return $1 per day overdue (0 if not overdue)

Helper Functions

- `generateId()` - already implemented
- `isOverdue(dueDate)` - check if date has passed
- `getDaysOverdue(dueDate)` - calculate overdue days
- `formatBookInfo(book)` - return formatted HTML string
- `createBookElement(book)` - create DOM element for a book
- `updateBookDisplay(bookId)` - update single book display
- `showStatus(message, isError)` - display status messages

3. DOM Interaction

- Use `document.getElementById()` to get input values
- Use `document.createElement()` and `appendChild()` to create UI elements
- Add event listeners with `addEventListener()`
- Update element content with `innerHTML` and `textContent`
- Handle user interactions through buttons and inputs

4. Event Handlers

Implement these functions to handle button clicks:

- `handleAddBook()` - get form values, call addBook, show status
- `handleRemoveBook()` - get ID, call removeBook, show status
- `handleSearchBooks()` - get query, search, display results
- `handleBorrowBook()` - get ID and name, call borrowBook, show status
- `handleReturnBook()` - get ID, call returnBook, show status

5. Quick Actions

Implement quick action functions for book cards:

- `quickBorrow(bookId)` - prompt for name, borrow book
- `quickReturn(bookId)` - return book directly
- `quickRemove(bookId)` - confirm and remove book

Sample Data

The library starts with 3 sample books:

```
const library = [
    { id: 1, title: "JavaScript Basics", author: "John Doe", year: 2020,
available: true },
    { id: 2, title: "Web Development", author: "Jane Smith", year: 2019,
available: false, borrower: "Ahmed", dueDate: "2024-01-15" },
    { id: 3, title: "Programming Fundamentals", author: "Bob Johnson", year: 2021,
available: true }
];
```