

Hands-on Lab: Develop an Application to Create Book Management System Using Objects



Estimated time needed: 30 minutes

What you will learn

In this lab, you will explore concepts such as collecting user input through HTML forms, handling input data using JavaScript objects, manipulating arrays to manage book data, dynamically update to display books, and implementing functionalities like adding, editing, and deleting book entries. This practical exercise provides learners with insights into fundamental concepts such as array methods, form handling, and user interaction, forming a foundational understanding of web development principles.

Learning objectives

After completing this lab, you will be able to:

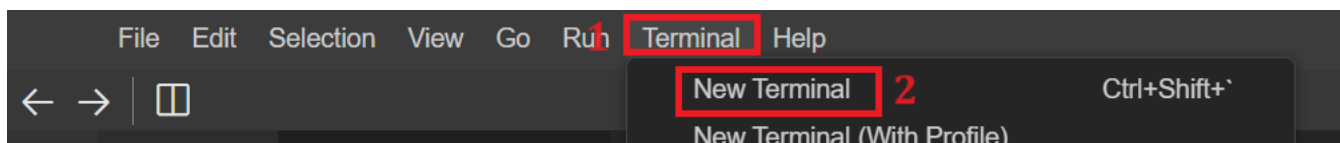
- **Implement a book management interface:** Create an interface to collect book details including name, author, description, and page count using HTML form elements.
- **Store and manage book data:** Use JavaScript arrays and objects to store and manage book information entered by users, allowing for functionalities like adding, editing, and deleting book entries.
- **Dynamic display of book information:** Dynamically update the webpage by displaying the entered book details in a structured format, enabling users to view a list of added books.
- **Interactive user experience:** Facilitate user interaction by allowing edits and deletions of book entries directly from the displayed list, enhancing the usability of the book management system.

Prerequisites

- Basic Knowledge of HTML and Git commands.
- Basic understanding of JavaScript variables and their scope.
- Web browser with a console (Chrome DevTools, Firefox Console, and so on).

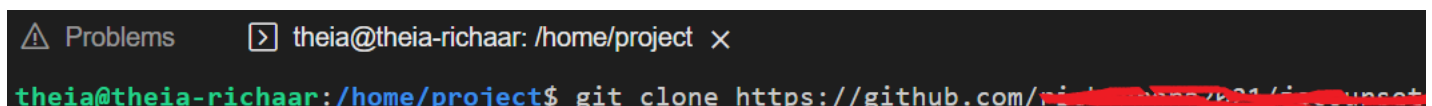
Step 1: Setting up the environment

1. Firstly, you need to clone your main repository in the **Skills Network Environment** which you have created in the first lab and where you have pushed all of your previous labs files and folders. Follow given steps:
 - For this click on terminal on the top-right window pane and then select **New Terminal**.

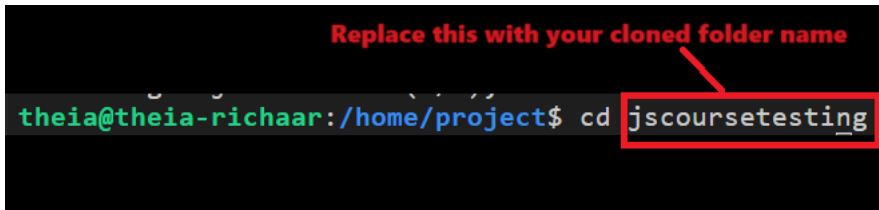


- Perform `git clone` command by writing given command in the terminal.
`git clone <github-repository-url>`

Note: Put your own GitHub repository link instead of `<github-repository-url>`.



- Above step will clone folder for your GitHub repository under project folder in explorer. You will also see multiple folders inside cloned folder.
- Now you need to navigate inside the cloned folder. For this write given command in the terminal:
`cd <repository-folder-name>`



Note: Write your cloned folder name instead of <repository-folder-name>. Perform `git clone` if you have logged out of **Skills Network Environment** and you cannot see any files or folder after you logged in.

- Now select **cloned Folder Name** folder, right-click on it and click on **New Folder**. Enter folder name as **bookSystem**. It will create the folder for you. Then select **bookSystem** folder, right-click and select **New File**. Enter file named as **book_system.html** and click OK. It will create your HTML file.
- Now select **bookSystem** folder again, right click and select **New File**. Enter file named as **book_system.js** and click OK. It will create your JavaScript file.
- Create basic template structure of the HTML file by adding the following code content.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Book Management System</title>
</head>
<body>
  <h1>Book Management System</h1>
  <label for="bookName">Book Name:</label><br>
  <input type="text" id="bookName"><br><br>
  <label for="authorName">Author Name:</label><br>
  <textarea id="authorName" rows="4"></textarea><br><br>
  <label for="bookDescription">Book Description:</label><br>
  <textarea id="bookDescription" rows="4"></textarea><br><br>
  <label for="pagesNumber">No. of Pages:</label><br>
  <input type="number" id="pagesNumber"><br><br>
  <button onclick="addBook()">Add Book</button>
  <h2>books</h2>
  <div id="books">
    <!-- books will be displayed here -->
  </div>
  <script src="./book_system.js"></script>
</body>
</html>
```

- **HTML structure:** The code creates a basic HTML structure for a Book Management System interface. It consists of various HTML elements, such as `<input>`, `<textarea>`, and `<button>` used to gather information about books, including the book name, author name, book description, and the number of pages. The interface provides labeled input fields and a button to add books to the system.
- **User input handling:** Each input field (book name, author name, description, and number of pages) is associated with an ID attribute, allowing JavaScript to retrieve their values. When the "Add Book" button is clicked, it triggers the `addBook()` function (not included in the provided code) to collect the entered book details from the input fields.
- **Book display section:** The code includes a section `<div id="books">` meant to display the added books. However, it lacks the JavaScript functionality to populate and display the books. As it stands, this section is indicated as the area where the books will be displayed, but there's no JavaScript logic provided to showcase the added books in this section.
- To include js file in **book_system.html**, a script tag is used in HTML file above `</body>` tag.

Note: After pasting the code, save your file.

Step 2: Defining variables and functions

- In **book_system.js**, declare an empty books array variable.

```
let books = [];
```

2. Create **addBook** function to add book details in the management system. For this, include the code provided below after the previous code.

```
function addBook() {
  const bookName = document.getElementById('bookName').value;
  const authorName = document.getElementById('authorName').value;
  const bookDescription = document.getElementById('bookDescription').value;
  const pageNumber = parseInt(document.getElementById('pageNumber').value);
  if (bookName && authorName && bookDescription && !isNaN(pageNumber)) {
    const book = {
      name: bookName,
      authorName: authorName,
      bookDescription: bookDescription,
      pageNumber: pageNumber
    };
    books.push(book);
    showbooks();
    clearInputs();
  } else {
    alert('Please fill in all fields correctly.');
```

- **Data collection and validation:** The function retrieves the values entered by the user in the HTML input fields for book name, author name, book description, and number of pages. It uses `document.getElementById('elementId').value` to access these values. The code then checks if all fields have been filled (`bookName`, `authorName`, `bookDescription`) and ensures that `pageNumber` is a valid number (using `!isNaN(pageNumber)`). If any field is empty or if `pageNumber` is not a valid number, an alert prompts the user to fill in all fields correctly.
- **Creating a book object:** Upon successful validation, the function creates a book object containing properties such as `name`, `authorName`, `bookDescription`, and `pageNumber`, storing the user-entered data. This book object is then pushed into the `books` array, effectively adding the new book to the collection.
- **Display and clear:** After adding the book, the `showbooks()` function is called to update the display with the newly added book details. Additionally, the `clearInputs()` function is invoked to reset all input fields, providing a clean interface for the user to add another book without manual clearing of previous inputs.

Step 3: Defining function to show books

1. Create a `showbooks` function to show book details in the management system. For this, include the given code in js file after the previous code.

```
function showbooks() {
  const booksDiv = books.map((book, index) => `<h1>book Number: ${index + 1}</h1>
  <p><strong>Book Name: </strong>${book.name}</p>
  <p><strong>Author Name:</strong> ${book.authorName}</p>
  <p><strong>Book Description:</strong> ${book.bookDescription}</p>
  <p><strong>No. of Pages:</strong> ${book.pageNumber} page(s)</p>
  <button onclick="editbook(${index})">Edit</button>`
  );
  document.getElementById('books').innerHTML = booksDiv.join('');
}
```

- The `map()` function iterates through the `books` array, creating a new array of HTML elements or strings based on the book information present in each array element.
- Generating book information:
 - For each book in the `books` array, it creates a structured HTML representation.
 - It constructs an HTML string that includes:
 - A heading displaying the book number `<h1>book Number: ${index + 1}</h1>`.
 - Paragraphs `<p>` containing details about the book such as name, author name, book description, and the number of pages.
 - One buttons for editing `<button onclick="editbook(${index})">Edit</button>` a specific book. This button is configured to call the `editbook()` function, passing the index of the book as a parameter.

- The `join("")` method is employed to concatenate all the HTML elements generated for each book into a single string. This string representation of HTML elements allows the content to be inserted as a single block of HTML.
- The `innerHTML` property of the HTML element with the ID 'books' is then set to the generated HTML string (`booksDiv`). This action replaces the existing content within the 'books' element with the newly created HTML structure representing the books.

2. Add the `editbook` function to populate form fields with the selected book's data:

```
function editbook(index) {
    const book = books[index];
    document.getElementById('bookName').value = book.name;
    document.getElementById('authorName').value = book.authorName;
    document.getElementById('bookDescription').value = book.bookDescription;
    document.getElementById('pagesNumber').value = book.pagesNumber;
    books.splice(index, 1); // Remove old entry
    showbooks(); // Refresh list
}
```

- `editbook` function allows users to edit a book's details by pre-filling the form with its existing data.
- `const book = books[index];` Fetches the book from the `books` array using the given index.
- **Populate Form Fields:**
 - Sets the book's name in the `bookName` input field.
 - Sets the author's name in the `authorName` input field.
 - Sets the book's description in the `bookDescription` input field.
 - Sets the number of pages in the `pagesNumber` input field.
- `books.splice(index, 1);` Deletes the selected book from the array using `splice(index, 1)` to prevent duplicate entries when saving changes.
- Calls `showbooks()` to update the book list and reflect changes.

Step 4: Defining function to clear books

1. Create a **`clearInputs`** function to clear the book details in the management system. For this include given code in the **`book_system.js`** file after the previous code.

```
function clearInputs() {
    document.getElementById('bookName').value = '';
    document.getElementById('authorName').value = '';
    document.getElementById('bookDescription').value = '';
    document.getElementById('pagesNumber').value = '';
}
```


- Above code clears the book details by assigning empty value to HTML element using ID's.

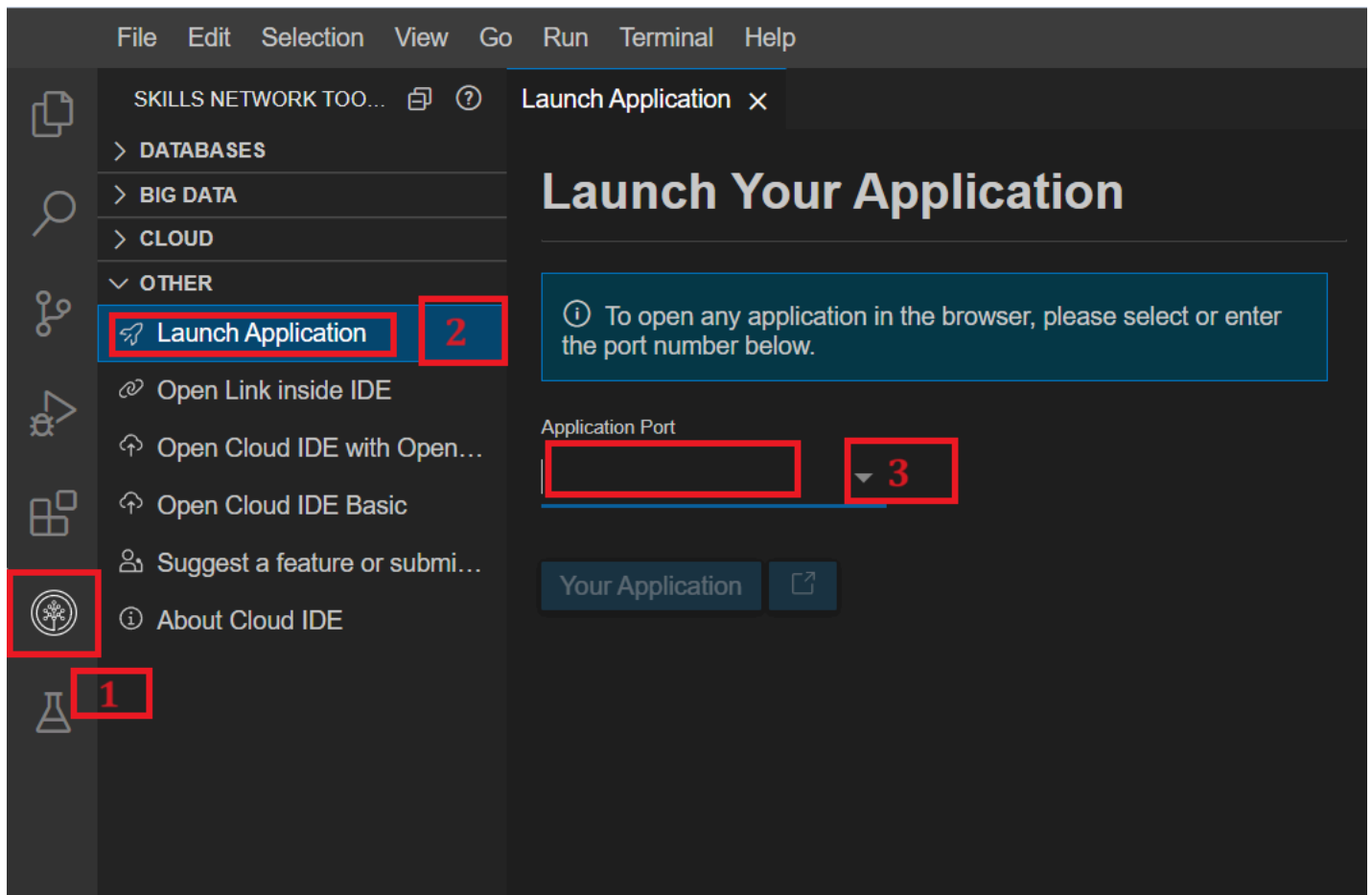
Step 5: Check the output

1. To view your HTML page, right-click the **`book_system.html`** file after selecting this file, then select "Open with Live Server".
2. The server should start on port 5500, indicated by a notification on the bottom right side.

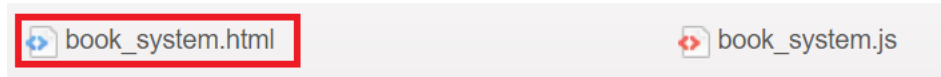


3. Click on the Skills Network button on the left (refer to number 1), it will open the "Skills Network Toolbox". Then click **Launch Application** (refer to number 2).

From there you enter the port no. as 5500 at number 3 and click on this button .



4. It will open your default browser where you will see **cloned-folder-name** folder name. Click on that **cloned-folder-name** folder name and then click on **bookSystem** folder name. You will see files related to this folder where you will click again on **book_system.html** file as shown below.



5. It will open the HTML page as shown below.

Book Management System

Book Name:

Author Name:

Book Description

No. of Pages

Add Book

6. Now enter the details and then click on **Add Book** button and then you will see the details of books. In the given screenshot, the entries have been submitted twice without refreshing the page.

book Number: 1

Book Name: Mathematics

Author Name: Example 1

Book Description: This book gives detailed information related to mathematics.

No. of Pages: 222 page(s)

Edit

book Number: 2

Book Name: JavaScript

Author Name: Example 2

Book Description: This book gives detailed information related to JavaScript.

No. of Pages: 420 page(s)

Edit

Note: After pasting the code, save your file. You can use any output method for saving. If you make further edits to your code, simply refresh your browser running on port number 5500. This eliminates the need to relaunch the application repeatedly.

Step 6: Perform Git commands

1. Perform `git add` to add latest files and folder by writing given command in terminal in git environment.

```
git add --a
```

Make sure terminal should have path as follows:



2. Then perform `git commit` in the terminal. While performing `git commit`, terminal can show message to set up your `git config --global` for user.name and user.email. If yes, then you need to perform `git config` command as well for user.name and user.email as given.

```
git config --global user.email "you@example.com"
```

```
git config --global user.name "Your Name"
```

Note: Replace data within quotes with your own details.

Then perform commit command as given:

```
git commit -m "message"
```

3. Then perform `git push` just by writing given command in terminal.

```
git push origin
```

- After the push command, the system will prompt you to enter your username and password. Enter the username for your GitHub account and the password that you created in the first lab. After entering the credentials, all of your latest folders and files will be pushed to your GitHub repository.

Practice task

1. In this practice task, you need to create a dynamic button at the time when user-entered details will be shown as output after clicking on **Add Book** button.
2. You need to create a delete button as shown below. For this you need to create a **deletebook** function, which can include the below given partial code.

```
<button onclick=".....">Delete</button>
```


book Number: 1

Book Name: Mathematics

Author Name: Example 1

Book Description: This book gives detailed information related to mathematics.

No. of Pages: 222 page(s)



3. The **deletebook** function will be called when the user clicks on the delete button. This function will also delete the book's detail in the management system for that particular ID.

Note: The particular ID is the array index number where the details of the particular book is stored.

4. To delete the book's detail you can use array method `splice()` inside **deletebook** function. This method is used to modify the books array, removing a single element starting from the specified index.

```
books.splice(index, 1);
```

5. This function deletes the book entry at the given index from the books array. After deletion, call the `showBooks` function to refresh the displayed book list.

► [Click here for solution code](#)

Summary

1. **Book management interface:** The code sets up a simple web interface for managing books. It includes input fields for book name, author name, book description, and the number of pages. Users can add books using the "Add Book" button.
2. **Dynamic display of books:** Upon adding books, the `showbooks()` function dynamically generates HTML content to display the list of added books. It formats and structures the book details using JavaScript's `map()` function to create HTML elements for each book, including buttons for editing and deleting individual book entries.
3. **Book handling functionality:** The code provides functionalities for editing and deleting book entries. The `editbook()` function allows users to edit book details by populating the input fields with the selected book's information. Meanwhile, the `deletebook()` function removes a book entry from the displayed list when the corresponding delete button is clicked. Both editing and deletion actions are handled by updating the book list dynamically using the `showbooks()` function, which refreshes the displayed list after any change.

© IBM Corporation. All rights reserved.