Steps to Start Writing the Code from Scratch

Here's how you can systematically approach building your Seat Booking System with the JavaScript functionality you've outlined.

---

1. Understand the Core Requirements

- **Seat Management**: Manage seat bookings and cancellations using closures.
- **Dynamic Rendering**: Display seat availability dynamically with appropriate styles.
- **User Interaction**: Allow users to book, cancel, and reset seat bookings.
- **Actions**: Confirm bookings and cancel all bookings.

---

2. Plan the Code Structure

Divide your functionality into these core modules:

- **Theater Setup**: Create and manage the seat matrix.
- **Seat Operations**: Handle booking, cancellation, and toggling of seat status.
- **Rendering**: Dynamically update the UI based on seat status.
- **User Actions**: Implement buttons for confirming and canceling bookings.

---

3. Set Up the Theater

- Use a closure-based `createTheater` function to manage the seat matrix.
- Initialize a 2D array of seats with properties such as `booked`.
- Provide methods for:
  - Booking a seat.
  - Canceling a seat.
  - Retrieving the current seat status.

---

4. Render Seats Dynamically

- Write a `renderSeats` function to:
  - Clear the existing DOM content.
  - Iterate over the seat matrix and create buttons for each seat.

    - Style buttons based on their booking status (e.g., green for available, red for booked).
    - Add event listeners to toggle seat status on button clicks.

---

5. Implement Seat Booking Logic

- Write a `toggleSeat` function to:
    - Check the seat’s current status.
    - Book or cancel the seat accordingly.
    - Re-render the seat layout to reflect changes.

---

6. Handle User Actions

- **Confirm Booking**:
    - Attach an event listener to the "Confirm Booking" button.
    - Display a confirmation alert or summary of booked seats.
- **Cancel All Bookings**:
    - Attach an event listener to the "Cancel All" button.
    - Iterate over the seat matrix and reset all seats to available.
    - Re-render the seat layout.

---

7. Test and Debug

- Test the following scenarios:
    - Initial rendering of seats.
    - Booking and canceling individual seats.
    - Confirming bookings.
    - Canceling all bookings.
- Debug issues using `console.log` and browser dev tools.

---

8. Optimize the Code

- Modularize the functionality into reusable functions.
- Add meaningful comments for better readability.
- Handle edge cases, such as invalid seat coordinates or booking an already booked seat.

Suggested Order to Write the Code

1. **Initialize Theater with Seat Matrix**
2. **Implement Seat Booking and Cancellation Logic**
3. **Render Seats Dynamically**
4. **Attach Event Listeners for User Actions**
5. **Test and Debug**

Tools to Assist

- **Console Logs**: Debug seat operations and rendering logic.
- **Browser DevTools**: Inspect DOM elements and verify event listeners.

By following this structured approach, you'll create a robust Seat Booking System with dynamic and interactive features. Let me know if you need further assistance!