Prerequisites for Developing a Seat Booking System

Below is a breakdown of the prerequisites and development workflow for implementing the provided seat booking logic using closures.

---

1. Technical Skills

JavaScript Concepts

1. **Closures**:
    - Encapsulate seat booking logic using `createTheater` to maintain state privately.
2. **Array Manipulation**:
    - Use `Array.from` to create a 2D array representing the seating arrangement.
    - Traverse and modify arrays dynamically with `forEach`.
3. **DOM Manipulation**:
    - Dynamically render seat buttons using `createElement` and manage their styles.
4. **Event Handling**:
    - Attach click events to each seat for toggling booking status.
    - Handle confirmation and cancellation events for all bookings.
5. **CSS Class Management**:
    - Dynamically update button styles (`bg-red-500`, `bg-green-500`) based on booking status.

---

2. Functional Requirements

Core Features

1. **Initialize Theater**:
    - Generate a seating arrangement with rows and columns, where each seat is initially `available`.

2. **Dynamic Rendering**:
    - Render seats in a grid layout with unique styles for `booked` and `available` seats.
3. **Toggle Seat Booking**:
    - Allow individual seats to be toggled between `booked` and `available` states.
4. **Confirm Booking**:
    - Simulate booking confirmation with a user alert.
5. **Cancel All Bookings**:
    - Reset all seats to `available` status dynamically.

---

3. UI Requirements

Seat Layout

- Render seats in a grid format.
- Each seat should:
    - Display its position (`row-col` format).
    - Have a distinct color for `booked` and `available` states.
    - Display a tooltip for its current state (`Booked` or `Available`).

Buttons

1. **Confirm Booking**:
    - A button to simulate booking confirmation.
2. **Cancel All Bookings**:
    - A button to reset all seats to `available`.

Styling

- Use CSS or a framework like **Tailwind CSS** for:
    - Button colors (`bg-green-500`, `bg-red-500`).
    - Layout alignment (`flex`, `grid`).

---

4. Key Functions

1. `createTheater(rows, cols)`

- Initialize a 2D array for seat states.
- Encapsulate `bookSeat`, `cancelSeat`, and `getSeats` methods for managing state.

2. `renderSeats()`

- Dynamically render seat buttons based on their state (`booked` or `available`).

3. `toggleSeat(row, col)`

- Toggle the booking status of a specific seat.
- Re-render the seat layout after updating state.

4. `confirmBooking`

- Display an alert or message to confirm the booking action.

5. `cancelAllBookings()`

- Iterate over all seats and reset them to `available`.
- Re-render the updated seat layout.

---

5. Development Workflow

1. **Setup the Theater State**:
    - Initialize the theater seating arrangement using `createTheater`.
2. **Render Seats Dynamically**:
    - Implement `renderSeats` to create buttons for each seat based on the current state.
3. **Toggle Seat Booking**:
    - Add click event listeners to toggle seat booking (`booked` ⇔ `available`).
4. **Implement Confirmation Logic**:
    - Add functionality to simulate booking confirmation.
5. **Reset All Bookings**:
    - Implement a reset mechanism to cancel all bookings at once.

6. **Error Handling**:
    - Handle edge cases, such as invalid seat indices (though unlikely with this implementation).

---

6. Testing Scenarios

Functional Tests

1. **Initialize Seats**:
    - Verify all seats are initialized as `available`.
2. **Toggle Booking**:
    - Test toggling individual seats between `booked` and `available`.
3. **Confirm Booking**:
    - Ensure the confirmation action does not alter seat states.
4. **Cancel All Bookings**:
    - Verify that all seats are reset to `available`.

Edge Cases

1. Clicking on seats multiple times consecutively.
2. Confirming or canceling bookings without any seats toggled.

---

7. Optional Enhancements

Seat Selection Summary

- Display the number of seats booked and available in real-time.

Pricing System

- Assign prices to seats and calculate the total cost of selected seats.

Persistent Storage

- Save the booking state using `localStorage` or a backend API.

Seat Categories

- Categorize seats (e.g., Regular, VIP) with different styles and pricing.

Accessibility

- Add focus indicators and keyboard navigation for seat selection.

---

8. Suggested Styling with Tailwind CSS

Seat Buttons

- **Available**: `bg-green-500 text-white`
- **Booked**: `bg-red-500 text-white`

Layout

- Use `grid grid-cols-10 gap-2` for seat arrangement.
- Wrap in a container with `flex flex-col items-center`.

By following these prerequisites and workflows, you can build a functional, user-friendly, and extensible seat booking system. Let me know if you'd like assistance with any part!