Esercizi per creare pagine web HTML5 e CSS3 che si concentrano su situazioni di vita reale. Ogni esercizio contiene una breve descrizione e il codice di esempio.

1. Lista della Spesa

Descrizione: Crea una pagina HTML che mostri una lista di articoli per la spesa. Gli articoli dovranno avere stili diversi per distinguere le categorie (frutta, verdura, carne, ecc.).

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Lista della Spesa</title>
    <style>
        body { font-family: Arial, sans-serif; }
        h2 { color: #4CAF50; }
        .frutta { color: red; }
        .verdura { color: green; }
        .carne { color: brown; }
    </style>
</head>
<body>
    <h2>Lista della Spesa</h2>
    <ul>
        <li class="frutta">Mele</li>
        <li class="verdura">Carote</li>
        <li class="carne">Pollo</li>
    </ul>
</body>
</html>
```

2. Curriculum Vitae

Descrizione: Crea una semplice pagina di curriculum vitae con sezioni per esperienza lavorativa, istruzione e competenze, con una struttura e uno stile chiari.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Curriculum Vitae</title>
    <style>
        body { font-family: Arial, sans-serif; line-height: 1.6; }
        h1 { color: #333; }
        section { margin-bottom: 20px; }
        h2 { color: #666; }
    </style>
</head>
<body>
    <h1>Mario Rossi</h1>
    <section>
        <h2>Esperienza Lavorativa</h2>
```

```
        <p>Front-end Developer presso WebTech (2019-2023)</p>
    </section>
    <section>
        <h2>Istruzione</h2>
        <p>Laurea in Informatica, Università di Pisa</p>
    </section>
    <section>
        <h2>Competenze</h2>
        <p>HTML, CSS, JavaScript</p>
    </section>
</body>
</html>
```

3. Form di Prenotazione Tavolo

Descrizione: Realizza un modulo HTML5 per la prenotazione di un tavolo al ristorante, completo di campi per nome, numero di persone, data e ora.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prenotazione Tavolo</title>
    <style>
        body { font-family: Arial, sans-serif; }
        form { max-width: 400px; margin: auto; }
        label, input { display: block; margin: 10px 0; }
    </style>
</head>
<body>
    <h2>Prenotazione Tavolo</h2>
    <form>
        <label for="nome">Nome:</label>
        <input type="text" id="nome" name="nome" required>
        <label for="persone">Numero di Persone:</label>
        <input type="number" id="persone" name="persone" min="1" required>
        <label for="data">Data:</label>
        <input type="date" id="data" name="data" required>
        <label for="ora">Ora:</label>
        <input type="time" id="ora" name="ora" required>
        <button type="submit">Prenota</button>
    </form>
</body>
</html>
```

4. Pagina di News

Descrizione: Crea una pagina con una sezione di news che mostra articoli recenti con titoli e descrizioni.

Codice:

```
<!DOCTYPE html>
<html lang="it">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>News</title>
    <style>
        body { font-family: Arial, sans-serif; }
        .article { margin-bottom: 20px; }
        h2 { color: #333; }
    </style>
</head>
<body>
    <h2>Ultime Notizie</h2>
    <div class="article">
        <h3>Notizia 1</h3>
        <p>Descrizione della prima notizia...</p>
    </div>
    <div class="article">
        <h3>Notizia 2</h3>
        <p>Descrizione della seconda notizia...</p>
    </div>
</body>
</html>
```

5. Pagina Portfolio Fotografia

Descrizione: Crea una galleria di immagini di un portfolio fotografico, con un layout in griglia.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Portfolio Fotografico</title>
    <style>
        body { font-family: Arial, sans-serif; }
        .gallery { display: grid; grid-template-columns: repeat(3, 1fr); gap:
10px; }
        .gallery img { width: 100%; height: auto; }
    </style>
</head>
<body>
    <h2>Portfolio Fotografico</h2>
    <div class="gallery">
        <img src="foto1.jpg" alt="Foto 1">
        <img src="foto2.jpg" alt="Foto 2">
        <img src="foto3.jpg" alt="Foto 3">
    </div>
</body>
</html>
```

6. Blog Personale

Descrizione: Una pagina blog con un layout semplice, mostrando articoli con titolo e testo.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog Personale</title>
    <style>
        body { font-family: Arial, sans-serif; }
        .post { border-bottom: 1px solid #ddd; padding: 10px 0; }
    </style>
</head>
<body>
    <h2>Blog Personale</h2>
    <div class="post">
        <h3>Titolo Post 1</h3>
        <p>Contenuto del post...</p>
    </div>
    <div class="post">
        <h3>Titolo Post 2</h3>
        <p>Contenuto del post...</p>
    </div>
</body>
</html>
```

7. Pagina di Contatto

Descrizione: Una pagina di contatto semplice con un modulo di contatto.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contattaci</title>
    <style>
        body { font-family: Arial, sans-serif; }
        form { max-width: 400px; margin: auto; }
    </style>
</head>
<body>
    <h2>Contattaci</h2>
    <form>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
        <label for="messaggio">Messaggio:</label>
        <textarea id="messaggio" name="messaggio" required></textarea>
        <button type="submit">Invia</button>
    </form>
</body>
</html>
```

8. Pagina Eventi

Descrizione: Un calendario di eventi con descrizione e data.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Eventi</title>
    <style>
        body { font-family: Arial, sans-serif; }
    </style>
</head>
<body>
    <h2>Eventi</h2>
    <div>
        <p><strong>Evento 1:</strong> 5 Dicembre 2024</p>
        <p>Descrizione...</p>
    </div>
    <div>
        <p><strong>Evento 2:</strong> 20 Dicembre 2024</p>
        <p>Descrizione...</p>
    </div>
</body>
</html>
```

1. Tabella Orari Lezioni

Descrizione: Crea una tabella che mostra gli orari di lezioni settimanali di una scuola, organizzati per giorno e ora.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Orario Lezioni</title>
    <style>
        table { width: 100%; border-collapse: collapse; }
        th, td { padding: 10px; border: 1px solid #ddd; text-align: center; }
        th { background-color: #f4f4f4; }
    </style>
</head>
<body>
    <h2>Orario Lezioni</h2>
    <table>
        <tr>
            <th>Giorno</th>
            <th>08:00 - 09:00</th>
            <th>09:00 - 10:00</th>
            <th>10:00 - 11:00</th>
        </tr>
        <tr>
            <td>Lunedì</td>
            <td>Matematica</td>
            <td>Italiano</td>
            <td>Storia</td>
        </tr>
```

```
        <tr>
            <td>Martedì</td>
            <td>Scienze</td>
            <td>Matematica</td>
            <td>Geografia</td>
        </tr>
    </table>
</body>
</html>
```

2. Lista Attività Giornaliere

Descrizione: Usa una lista per mostrare le attività giornaliere organizzate per ora e descrizione.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Attività Giornaliere</title>
    <style>
        ul { list-style-type: none; padding: 0; }
        li { padding: 10px; margin-bottom: 5px; background-color: #f4f4f4; }
        .ora { font-weight: bold; }
    </style>
</head>
<body>
    <h2>Attività Giornaliere</h2>
    <ul>
        <li><span class="ora">08:00</span> - Colazione</li>
        <li><span class="ora">09:00</span> - Lavoro</li>
        <li><span class="ora">13:00</span> - Pranzo</li>
        <li><span class="ora">18:00</span> - Palestra</li>
    </ul>
</body>
</html>
```

3. Tabella di Prezzi Prodotti

Descrizione: Crea una tabella per un listino prezzi, con colonne per prodotto, descrizione e prezzo.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prezzi Prodotti</title>
    <style>
        table { width: 80%; margin: auto; border-collapse: collapse; }
        th, td { padding: 10px; border: 1px solid #ddd; }
        th { background-color: #f8f8f8; }
    </style>
</head>
```

```
<body>
    <h2>Listino Prezzi</h2>
    <table>
        <tr>
            <th>Prodotto</th>
            <th>Descrizione</th>
            <th>Prezzo (€)</th>
        </tr>
        <tr>
            <td>Pane</td>
            <td>Pane fresco</td>
            <td>2.50</td>
        </tr>
        <tr>
            <td>Latte</td>
            <td>Latte intero 1L</td>
            <td>1.20</td>
        </tr>
    </table>
</body>
</html>
```

4. Lista di Obiettivi Annuali

Descrizione: Crea una lista non ordinata con gli obiettivi dell'anno.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Obiettivi Annuali</title>
    <style>
        ul { list-style-type: square; padding: 0; }
        li { padding: 10px; background-color: #e8f4f8; margin-bottom: 5px; }
    </style>
</head>
<body>
    <h2>Obiettivi Annuali</h2>
    <ul>
        <li>Risparmiare il 10% del reddito</li>
        <li>Visitare almeno 3 nuovi paesi</li>
        <li>Leggere 12 libri</li>
    </ul>
</body>
</html>
```

5. Tabella di Prenotazioni Hotel

Descrizione: Una tabella per visualizzare le prenotazioni di un hotel con nome ospite, data di arrivo e di partenza.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prenotazioni Hotel</title>
    <style>
        table { width: 70%; border-collapse: collapse; }
        th, td { padding: 8px; text-align: left; border: 1px solid #ddd; }
        th { background-color: #f2f2f2; }
    </style>
</head>
<body>
    <h2>Prenotazioni Hotel</h2>
    <table>
        <tr>
            <th>Nome Ospite</th>
            <th>Data di Arrivo</th>
            <th>Data di Partenza</th>
        </tr>
        <tr>
            <td>Mario Rossi</td>
            <td>2024-12-01</td>
            <td>2024-12-10</td>
        </tr>
        <tr>
            <td>Giulia Verdi</td>
            <td>2024-12-03</td>
            <td>2024-12-08</td>
        </tr>
    </table>
</body>
</html>
```

6. Lista di Ricette

Descrizione: Crea una lista di ricette, ogni elemento contiene il nome della ricetta e gli ingredienti come lista annidata.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ricette</title>
    <style>
        ul { padding: 0; }
        li { margin-bottom: 10px; }
        .ingredienti { list-style-type: circle; padding-left: 20px; }
    </style>
</head>
<body>
    <h2>Ricette</h2>
    <ul>
        <li>
            <strong>Pasta al Pomodoro</strong>
            <ul class="ingredienti">
```

```
                <li>Pasta</li>
                <li>Pomodoro</li>
                <li>Basilico</li>
            </ul>
        </li>
        <li>
            <strong>Tiramisù</strong>
            <ul class="ingredienti">
                <li>Mascarpone</li>
                <li>Caffè</li>
                <li>Biscotti Savoiardi</li>
            </ul>
        </li>
    </ul>
</body>
</html>
```

7. Tabella Spese Mensili

Descrizione: Visualizza una tabella con le spese mensili per categorie (alimentari, trasporti, ecc.).

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Spese Mensili</title>
    <style>
        table { width: 60%; border-collapse: collapse; }
        th, td { padding: 8px; text-align: left; border: 1px solid #ddd; }
        th { background-color: #f2f2f2; }
    </style>
</head>
<body>
    <h2>Spese Mensili</h2>
    <table>
        <tr>
            <th>Categoria</th>
            <th>Importo (€)</th>
        </tr>
        <tr>
            <td>Alimentari</td>
            <td>200</td>
        </tr>
        <tr>
            <td>Trasporti</td>
            <td>100</td>
        </tr>
        <tr>
            <td>Intrattenimento</td>
            <td>80</td>
        </tr>
    </table>
</body>
</html>
```

8. Lista delle Cose da Fare (To-Do List)

Descrizione: Crea una lista di cose da fare organizzata con voci principali e secondarie (es. pulizia della casa con sottovoci per le stanze).

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cose da Fare</title>
    <style>
        ul { list-style-type: none; padding: 0; }
        .categoria { font-weight: bold; margin-top: 10px; }
        .sottovoci { list-style-type: circle; padding-left: 20px; }
    </style>
</head>
<body>
    <h2>Cose da Fare</h2>
    <ul>
        <li class="categoria">Pulizie</li>
        <ul class="sottovoci">
            <li>Cucina</li>
            <li>Bagno</li>
        </ul>
        <li class="categoria">Lavoro</li>
        <ul class="sottovoci">
            <li>Riassumere documenti</li>
            <li>Rispondere alle email</li>
        </ul>
    </ul>
</body>
</html>
```

9. Tabella dei Corsi Universitari

Descrizione: Una tabella che elenca i corsi universitari con nome, docente e numero di crediti.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Corsi Universitari</title>
    <style>
        table { width: 80%; border-collapse: collapse; margin: auto; }
        th, td { padding: 10px; border: 1px solid #ddd; }
        th { background-color: #f4f4f4; }
    </style>
</head>
<body>
    <h2>Corsi Universitari</h2>
    <table>
        <tr>
            <th>Nome Corso</th>
            <th>Docente</th>
```

```
            <th>Crediti</th>
        </tr>
        <tr>
            <td>Matematica 1</td>
            <td>Prof. Verdi</td>
            <td>6</td>
        </tr>
        <tr>
            <td>Fisica</td>
            <td>Prof. Bianchi</td>
            <td>8</td>
        </tr>
    </table>
</body>
</html>
```

10. Tabella di Comparazione Prodotti

Descrizione: Una tabella di comparazione tra prodotti, con specifiche tecniche come nome, modello e prezzo.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Comparazione Prodotti</title>
    <style>
        table { width: 100%; border-collapse: collapse; }
        th, td { padding: 10px; border: 1px solid #ddd; text-align: center; }
        th { background-color: #e0e0e0; }
    </style>
</head>
<body>
    <h2>Comparazione Prodotti</h2>
    <table>
        <tr>
            <th>Nome</th>
            <th>Modello</th>
            <th>Prezzo (€)</th>
        </tr>
        <tr>
            <td>Prodotto A</td>
            <td>Modello 123</td>
            <td>300</td>
        </tr>
        <tr>
            <td>Prodotto B</td>
            <td>Modello 456</td>
            <td>450</td>
        </tr>
    </table>
</body>
</html>
```

11. Lista dei Libri Preferiti

Descrizione: Una lista di libri preferiti organizzata per autore e titolo.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Libri Preferiti</title>
    <style>
        ul { padding: 0; list-style-type: none; }
        li { margin-bottom: 10px; }
        .titolo { font-weight: bold; }
    </style>
</head>
<body>
    <h2>Libri Preferiti</h2>
    <ul>
        <li><span class="titolo">1984</span> di George Orwell</li>
        <li><span class="titolo">Il Signore degli Anelli</span> di J.R.R.
Tolkien</li>
    </ul>
</body>
</html>
```

12. Tabella delle Temperature Settimanali

Descrizione: Una tabella che mostra le temperature minime e massime della settimana per una città.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Temperature Settimanali</title>
    <style>
        table { width: 60%; margin: auto; border-collapse: collapse; }
        th, td { padding: 10px; border: 1px solid #ddd; text-align: center; }
        th { background-color: #d0e4f7; }
    </style>
</head>
<body>
    <h2>Temperature Settimanali</h2>
    <table>
        <tr>
            <th>Giorno</th>
            <th>Minima (°C)</th>
            <th>Massima (°C)</th>
        </tr>
        <tr>
            <td>Lunedì</td>
            <td>10</td>
            <td>18</td>
        </tr>
        <tr>
            <td>Martedì</td>
```

```
            <td>11</td>
            <td>20</td>
        </tr>
    </table>
</body>
</html>
```

13. Lista delle Attrezzature Sportive

Descrizione: Crea una lista di attrezzature sportive necessarie per varie attività.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Attrezzature Sportive</title>
    <style>
        ul { padding: 0; list-style-type: none; }
        .attrezzatura { font-weight: bold; margin-bottom: 5px; }
        .descrizione { margin-left: 15px; }
    </style>
</head>
<body>
    <h2>Attrezzature Sportive</h2>
    <ul>
        <li>
            <span class="attrezzatura">Pallone</span>
            <p class="descrizione">Per il calcio e il basket.</p>
        </li>
        <li>
            <span class="attrezzatura">Racchetta</span>
            <p class="descrizione">Per il tennis e il badminton.</p>
        </li>
    </ul>
</body>
</html>
```

14. Tabella dei Turni di Lavoro

Descrizione: Crea una tabella che mostra i turni di lavoro di diversi dipendenti.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Turni di Lavoro</title>
    <style>
        table { width: 80%; border-collapse: collapse; }
        th, td { padding: 10px; border: 1px solid #ddd; text-align: center; }
        th { background-color: #f4f4f4; }
    </style>
```

```
</head>
<body>
    <h2>Turni di Lavoro</h2>
    <table>
        <tr>
            <th>Dipendente</th>
            <th>Giorno</th>
            <th>Turno</th>
        </tr>
        <tr>
            <td>Mario Rossi</td>
            <td>Lunedì</td>
            <td>Mattina</td>
        </tr>
        <tr>
            <td>Anna Bianchi</td>
            <td>Martedì</td>
            <td>Pomeriggio</td>
        </tr>
    </table>
</body>
</html>
```

15-20: Altri Esercizi Simili

- **Tabella di Pianificazione Settimanale**: Pianifica la tua settimana, giorno per giorno.
- **Lista di Film da Guardare**: Organizza i film in lista, con il genere.
- **Tabella degli Ospiti Evento**: Elenco dei partecipanti con nome e conferma.
- **Lista dei Contatti Importanti**: Lista di contatti con ruoli e numeri di telefono.
- **Tabella Calorie per Alimenti**: Una tabella per il conteggio delle calorie.
- **Lista delle Spese Mensili**: Elenco delle principali spese di ogni mese.

15. Tabella di Pianificazione Settimanale

Descrizione: Una tabella per pianificare le attività settimanali, con colonne per ogni giorno e righe per mattina, pomeriggio, e sera.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pianificazione Settimanale</title>
    <style>
        table { width: 100%; border-collapse: collapse; }
        th, td { padding: 8px; border: 1px solid #ddd; text-align: center; }
        th { background-color: #f4f4f4; }
    </style>
</head>
<body>
    <h2>Pianificazione Settimanale</h2>
    <table>
        <tr>
```

```
            <th>Orario</th>
            <th>Lunedì</th>
            <th>Martedì</th>
            <th>Mercoledì</th>
            <th>Giovedì</th>
            <th>Venerdì</th>
        </tr>
        <tr>
            <td>Mattina</td>
            <td>Palestra</td>
            <td>Lezione di inglese</td>
            <td>Yoga</td>
            <td>Lavoro di gruppo</td>
            <td>Studio</td>
        </tr>
        <tr>
            <td>Pomeriggio</td>
            <td>Studio</td>
            <td>Progetto</td>
            <td>Studio</td>
            <td>Biblioteca</td>
            <td>Riposo</td>
        </tr>
        <tr>
            <td>Sera</td>
            <td>Cena con amici</td>
            <td>Film</td>
            <td>Riposo</td>
            <td>Allenamento</td>
            <td>Relax</td>
        </tr>
    </table>
</body>
</html>
```

---

16. Lista di Film da Guardare

Descrizione: Crea una lista di film da guardare, organizzata per genere.

Codice:

```
html
Copia codice
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Film da Guardare</title>
    <style>
        ul { padding: 0; }
        .genere { font-weight: bold; margin-top: 10px; }
        .film-list { list-style-type: circle; padding-left: 20px; }
    </style>
</head>
<body>
    <h2>Film da Guardare</h2>
    <ul>
        <li class="genere">Azione</li>
        <ul class="film-list">
```

```
            <li>Mad Max: Fury Road</li>
            <li>John Wick</li>
        </ul>
        <li class="genere">Commedia</li>
        <ul class="film-list">
            <li>Forrest Gump</li>
            <li>The Grand Budapest Hotel</li>
        </ul>
    </ul>
</body>
</html>
```

17. Tabella degli Ospiti Evento

Descrizione: Una tabella per gestire la lista degli ospiti di un evento, con nome e stato di conferma.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ospiti Evento</title>
    <style>
        table { width: 70%; margin: auto; border-collapse: collapse; }
        th, td { padding: 10px; border: 1px solid #ddd; text-align: left; }
        th { background-color: #f8f8f8; }
    </style>
</head>
<body>
    <h2>Lista degli Ospiti</h2>
    <table>
        <tr>
            <th>Nome</th>
            <th>Confermato</th>
        </tr>
        <tr>
            <td>Lucia Verdi</td>
            <td>Sì</td>
        </tr>
        <tr>
            <td>Marco Neri</td>
            <td>No</td>
        </tr>
        <tr>
            <td>Alessandro Bianchi</td>
            <td>Sì</td>
        </tr>
    </table>
</body>
</html>
```

18. Lista dei Contatti Importanti

Descrizione: Una lista di contatti con il ruolo e numero di telefono.

Codice:

```
html
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contatti Importanti</title>
    <style>
        ul { padding: 0; list-style-type: none; }
        .contatto { margin-bottom: 15px; }
        .ruolo { font-weight: bold; }
    </style>
</head>
<body>
    <h2>Contatti Importanti</h2>
    <ul>
        <li class="contatto">
            <span class="ruolo">Dottore</span> - 123-4567890
        </li>
        <li class="contatto">
            <span class="ruolo">Avvocato</span> - 098-7654321
        </li>
        <li class="contatto">
            <span class="ruolo">Parrucchiere</span> - 111-2223334
        </li>
    </ul>
</body>
</html>
```

19. Tabella Calorie per Alimenti

Descrizione: Una tabella per tenere traccia delle calorie degli alimenti, con nome e calorie per porzione.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Calorie per Alimenti</title>
    <style>
        table { width: 60%; margin: auto; border-collapse: collapse; }
        th, td { padding: 10px; border: 1px solid #ddd; text-align: center; }
        th { background-color: #f0f0f0; }
    </style>
</head>
<body>
    <h2>Calorie per Alimenti</h2>
    <table>
        <tr>
            <th>Alimento</th>
            <th>Calorie per Porzione</th>
        </tr>
        <tr>
            <td>Mela</td>
            <td>95</td>
```

```
        </tr>
        <tr>
            <td>Banana</td>
            <td>105</td>
        </tr>
        <tr>
            <td>Pane Integrale</td>
            <td>70</td>
        </tr>
    </table>
</body>
</html>
```

20. Lista delle Spese Mensili

Descrizione: Crea una lista delle spese mensili principali, con descrizione e importo.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Spese Mensili</title>
    <style>
        ul { padding: 0; list-style-type: none; }
        .spesa { display: flex; justify-content: space-between; background-
color: #f9f9f9; padding: 8px; margin-bottom: 5px; }
        .descrizione { font-weight: bold; }
        .importo { color: #333; }
    </style>
</head>
<body>
    <h2>Spese Mensili</h2>
    <ul>
        <li class="spesa">
            <span class="descrizione">Affitto</span>
            <span class="importo">€ 500</span>
        </li>
        <li class="spesa">
            <span class="descrizione">Bollette</span>
            <span class="importo">€ 150</span>
        </li>
        <li class="spesa">
            <span class="descrizione">Alimentari</span>
            <span class="importo">€ 200</span>
        </li>
    </ul>
</body>
</html>
```

Esercizio 1: Form di Prenotazione di una Sala Conferenze

Descrizione: Crea un form per prenotare una sala conferenze. Il form chiederà il nome del richiedente, il numero di partecipanti, la data e l'ora dell'evento. Ci saranno tre pulsanti: "Conferma" per inviare i dati, "Annulla" per resettare i campi e "Cancella Prenotazione" per eliminare eventuali dati già memorizzati.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prenotazione Sala Conferenze</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        form { max-width: 400px; margin: auto; padding: 20px; border: 1px solid
#ddd; border-radius: 5px; }
        label, input { display: block; width: 100%; margin-bottom: 10px; }
        button { margin-top: 10px; padding: 8px 12px; font-size: 16px; }
    </style>
</head>
<body>
    <h2>Prenotazione Sala Conferenze</h2>
    <form id="bookingForm">
        <label for="name">Nome:</label>
        <input type="text" id="name" required>

        <label for="participants">Numero Partecipanti:</label>
        <input type="number" id="participants" required>

        <label for="date">Data:</label>
        <input type="date" id="date" required>

        <label for="time">Ora:</label>
        <input type="time" id="time" required>

        <button type="button" onclick="confirmBooking()">Conferma</button>
        <button type="button" onclick="resetForm()">Annulla</button>
        <button type="button" onclick="clearData()">Cancella
Prenotazione</button>
    </form>

    <script>
        function confirmBooking() {
            // Ottiene i valori inseriti dall'utente
            let name = document.getElementById("name").value;
            let participants = document.getElementById("participants").value;
            let date = document.getElementById("date").value;
            let time = document.getElementById("time").value;

            // Verifica che tutti i campi siano completi
            if (name && participants && date && time) {
                alert("Prenotazione confermata per " + name + " con " +
participants + " partecipanti il " + date + " alle ore " + time + ".");
            } else {
                alert("Compila tutti i campi prima di confermare.");
            }
        }

        function resetForm() {
            // Rimuove tutti i valori dal form
            document.getElementById("bookingForm").reset();
        }

        function clearData() {
            // Conferma l'azione e resetta i dati se l'utente conferma
            if (confirm("Sei sicuro di voler cancellare la prenotazione?")) {
```

```
                resetForm();
                alert("Prenotazione cancellata.");
            }
        }
    </script>
</body>
</html>
```

Spiegazione JavaScript:

1. **`confirmBooking()`**:
    - Raccoglie i valori inseriti dall'utente usando `document.getElementById()`.
    - Se tutti i campi sono riempiti, mostra un messaggio di conferma con i dati inseriti; altrimenti, mostra un avviso.
2. **`resetForm()`**:
    - Utilizza il metodo `.reset()` sul form per cancellare tutti i campi.
3. **`clearData()`**:
    - Chiede conferma all'utente con `confirm()`.
    - Se l'utente conferma, richiama `resetForm()` e mostra un messaggio di conferma.

Esercizio 2: Calcolatrice Semplice per il Budget Mensile

Descrizione: Crea un form per calcolare il budget mensile, con i campi per entrate e spese e tre pulsanti: “Calcola” per fare i calcoli, “Azzera” per resettare il form e “Cancella” per confermare la cancellazione dei dati.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Calcolatrice Budget Mensile</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        form { max-width: 400px; margin: auto; padding: 20px; border: 1px solid
#ddd; border-radius: 5px; }
        label, input { display: block; width: 100%; margin-bottom: 10px; }
        button { margin-top: 10px; padding: 8px 12px; font-size: 16px; }
    </style>
</head>
<body>
    <h2>Calcolatrice Budget Mensile</h2>
    <form id="budgetForm">
        <label for="income">Entrate (€):</label>
        <input type="number" id="income" required>

        <label for="expenses">Spese (€):</label>
        <input type="number" id="expenses" required>

        <button type="button" onclick="calculateBudget()">Calcola</button>
        <button type="button" onclick="resetForm()">Azzera</button>
        <button type="button" onclick="clearBudget()">Cancella</button>
    </form>
```

```
    <script>
        function calculateBudget() {
            let income = parseFloat(document.getElementById("income").value);
            let expenses =
parseFloat(document.getElementById("expenses").value);

            if (!isNaN(income) && !isNaN(expenses)) {
                let balance = income - expenses;
                alert("Il bilancio mensile è: €" + balance.toFixed(2));
            } else {
                alert("Inserisci valori validi per entrate e spese.");
            }
        }

        function resetForm() {
            document.getElementById("budgetForm").reset();
        }

        function clearBudget() {
            if (confirm("Sei sicuro di voler cancellare il budget?")) {
                resetForm();
                alert("Dati del budget cancellati.");
            }
        }
    </script>
</body>
</html>
```

Spiegazione JavaScript:

1. **`calculateBudget()`**:
   - Converte i valori di entrate e spese in numeri usando `parseFloat`.
   - Calcola il bilancio sottraendo le spese dalle entrate e lo mostra.
2. **`resetForm()`**:
   - Reset dei campi.
3. **`clearBudget()`**:
   - Chiede conferma e resetta i dati in caso di risposta affermativa.

Esercizio 3: Checklist Spesa Settimanale

Descrizione: Una checklist per la spesa settimanale con pulsanti per selezionare e deselezionare tutte le voci. Utile per chi vuole prepararsi al supermercato.

Codice:

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Checklist Spesa</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        .item { margin: 10px 0; }
        button { margin-top: 10px; padding: 8px 12px; font-size: 16px; }
```

```
    </style>
</head>
<body>
    <h2>Checklist Spesa</h2>
    <div id="checklist">
        <div class="item"><input type="checkbox" class="grocery-item">
Latte</div>
        <div class="item"><input type="checkbox" class="grocery-item">
Pane</div>
        <div class="item"><input type="checkbox" class="grocery-item">
Uova</div>
        <div class="item"><input type="checkbox" class="grocery-item">
Frutta</div>
        <div class="item"><input type="checkbox" class="grocery-item">
Verdura</div>
    </div>

    <button onclick="selectAll()">Seleziona Tutti</button>
    <button onclick="deselectAll()">Deseleziona Tutti</button>

    <script>
        function selectAll() {
            // Seleziona tutti gli elementi della checklist
            let items = document.querySelectorAll(".grocery-item");
            items.forEach(item => item.checked = true);
        }

        function deselectAll() {
            // Deseleziona tutti gli elementi della checklist
            let items = document.querySelectorAll(".grocery-item");
            items.forEach(item => item.checked = false);
        }
    </script>
</body>
</html>
```

Spiegazione JavaScript:

1. **`selectAll()`**:
    - Usa `querySelectorAll` per selezionare tutte le checkbox.
    - Imposta `checked = true` su ciascuna, selezionando tutti gli elementi.
2. **`deselectAll()`**:
    - Stessa selezione di checkbox.
    - Imposta `checked = false` per deselezionare tutti.

Descrizione del Problema

Realizzeremo una pagina HTML con un form che permette di inserire i dati di una lista contatti (Nome, Cognome, Email e Telefono). Ogni volta che vengono inseriti nuovi dati, questi vengono visualizzati in una tabella che cresce dinamicamente. Inoltre, si potranno **modificare** i dati già inseriti, **eliminarli** e, se necessario, **aggiungerne di nuovi**. I dati verranno salvati in una matrice JavaScript, simulando una piccola gestione di database lato client.

Codice Completo

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Gestione Contatti</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        form { margin-bottom: 20px; }
        input, button { padding: 8px; margin: 5px; }
        table { width: 100%; border-collapse: collapse; margin-top: 20px; }
        th, td { border: 1px solid #ddd; padding: 10px; text-align: left; }
        th { background-color: #f4f4f4; }
    </style>
</head>
<body>
    <h2>Gestione Contatti</h2>

    <!-- Form di input per inserire i dati -->
    <form id="contactForm">
        <input type="text" id="name" placeholder="Nome" required>
        <input type="text" id="surname" placeholder="Cognome" required>
        <input type="email" id="email" placeholder="Email" required>
        <input type="text" id="phone" placeholder="Telefono" required>
        <button type="button" onclick="addContact()">Aggiungi Contatto</button>
    </form>

    <!-- Tabella per visualizzare i dati -->
    <table>
        <thead>
            <tr>
                <th>Nome</th>
                <th>Cognome</th>
                <th>Email</th>
                <th>Telefono</th>
                <th>Azioni</th>
            </tr>
        </thead>
        <tbody id="contactTable">
            <!-- Le righe della tabella saranno generate dinamicamente -->
        </tbody>
    </table>

    <script>
        // Array per memorizzare i dati dei contatti
        let contacts = [];

        // Funzione per aggiungere un nuovo contatto
        function addContact() {
            // Ottiene i valori dai campi di input
            let name = document.getElementById("name").value;
            let surname = document.getElementById("surname").value;
            let email = document.getElementById("email").value;
            let phone = document.getElementById("phone").value;

            // Crea un oggetto per il nuovo contatto e lo aggiunge alla matrice
            let contact = { name, surname, email, phone };
            contacts.push(contact);

            // Aggiorna la tabella e resetta i campi
```

```
    updateTable();
    document.getElementById("contactForm").reset();
}

// Funzione per aggiornare la tabella con i dati dell'array
function updateTable() {
    let table = document.getElementById("contactTable");
    table.innerHTML = "";  // Pulisce la tabella

    // Genera una nuova riga per ogni contatto nella matrice
    contacts.forEach((contact, index) => {
        let row = table.insertRow();
        row.insertCell(0).innerText = contact.name;
        row.insertCell(1).innerText = contact.surname;
        row.insertCell(2).innerText = contact.email;
        row.insertCell(3).innerText = contact.phone;

        // Crea i pulsanti di modifica ed elimina
        let actionsCell = row.insertCell(4);
        actionsCell.innerHTML = `
            <button onclick="editContact(${index})">Modifica</button>
            <button onclick="deleteContact(${index})">Elimina</button>
        `;
    });
}

// Funzione per eliminare un contatto
function deleteContact(index) {
    // Rimuove il contatto dall'array usando l'indice
    contacts.splice(index, 1);

    // Aggiorna la tabella
    updateTable();
}

// Funzione per modificare un contatto
function editContact(index) {
    // Ottiene i dati del contatto selezionato
    let contact = contacts[index];

    // Pre-popola i campi di input con i dati esistenti
    document.getElementById("name").value = contact.name;
    document.getElementById("surname").value = contact.surname;
    document.getElementById("email").value = contact.email;
    document.getElementById("phone").value = contact.phone;

    // Aggiorna il pulsante per confermare le modifiche
    document.getElementById("contactForm").onsubmit = function(event) {
        event.preventDefault();  // Evita il comportamento predefinito
        updateContact(index);
    };
}

// Funzione per confermare le modifiche di un contatto
function updateContact(index) {
    // Aggiorna i dati del contatto con i valori nei campi di input
    contacts[index] = {
        name: document.getElementById("name").value,
        surname: document.getElementById("surname").value,
        email: document.getElementById("email").value,
        phone: document.getElementById("phone").value
    };
```

```
            // Resetta il form e aggiorna la tabella
            document.getElementById("contactForm").reset();
            document.getElementById("contactForm").onsubmit = function(event) {
                event.preventDefault();
                addContact();
            };
            updateTable();
        }
    </script>
</body>
</html>
```

Spiegazione del Codice JavaScript

1. **`let contacts = [];`**: Dichiarazione di un array vuoto `contacts` che memorizzerà tutti i contatti come oggetti.
2. **Funzione `addContact()`**:
   - Raccoglie i valori inseriti nei campi di input per Nome, Cognome, Email e Telefono.
   - Crea un oggetto contatto `{ name, surname, email, phone }` e lo aggiunge all'array `contacts`.
   - Chiama `updateTable()` per aggiornare la visualizzazione dei dati nella tabella.
   - Esegue il reset dei campi del form dopo l'inserimento dei dati.
3. **Funzione `updateTable()`**:
   - Cancella tutto il contenuto della tabella (`table.innerHTML = "";`).
   - Cicla su ogni elemento dell'array `contacts` e per ciascuno di essi:
     - Crea una nuova riga nella tabella.
     - Popola le celle della riga con i dati del contatto (Nome, Cognome, Email, Telefono).
     - Aggiunge due pulsanti per ogni contatto: uno per modificare e uno per eliminare, associando a ciascun pulsante la funzione `editContact(index)` o `deleteContact(index)`.
4. **Funzione `deleteContact(index)`**:
   - Rimuove il contatto selezionato dalla matrice `contacts` usando `splice()`.
   - Chiama `updateTable()` per aggiornare la tabella dopo l'eliminazione.
5. **Funzione `editContact(index)`**:
   - Estrae l'oggetto `contact` dalla matrice `contacts` utilizzando `index`.
   - Inserisce i valori del contatto selezionato nei campi di input per consentire la modifica.
   - Modifica la funzione di `onsubmit` del form per chiamare `updateContact(index)` invece di `addContact()`, permettendo così di salvare le modifiche anziché aggiungere un nuovo contatto.
6. **Funzione `updateContact(index)`**:
   - Aggiorna l'oggetto `contact` in `contacts[index]` con i nuovi valori inseriti.
   - Reset del form e ripristino della funzione `onsubmit` del form per chiamare nuovamente `addContact()`.
   - Chiama `updateTable()` per riflettere le modifiche nella tabella.

Tre esercizi di difficoltà crescente che implementano tutte le operazioni CRUD (Create, Read, Update, Delete), ciascuno con una descrizione dettagliata del problema e una spiegazione riga per riga del codice.

Esercizio 1: Gestione di una Lista di Libri (CRUD di Base)

Descrizione: Realizzeremo una pagina HTML per gestire una piccola biblioteca virtuale. Ogni libro ha un Titolo, un Autore e un Anno di Pubblicazione. L'utente può aggiungere nuovi libri, visualizzarli in una tabella, modificarli e cancellarli. I dati saranno salvati in un array di oggetti in JavaScript.

Codice Completo:

```
html
Copia codice
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Lista Libri</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        form { margin-bottom: 20px; }
        input, button { padding: 8px; margin: 5px; }
        table { width: 100%; border-collapse: collapse; margin-top: 20px; }
        th, td { border: 1px solid #ddd; padding: 10px; text-align: left; }
        th { background-color: #f4f4f4; }
    </style>
</head>
<body>
    <h2>Gestione Libreria</h2>

    <!-- Form di input per inserire i dati del libro -->
    <form id="bookForm">
        <input type="text" id="title" placeholder="Titolo" required>
        <input type="text" id="author" placeholder="Autore" required>
        <input type="number" id="year" placeholder="Anno" required>
        <button type="button" onclick="addBook()">Aggiungi Libro</button>
    </form>

    <!-- Tabella per visualizzare i dati -->
    <table>
        <thead>
            <tr>
                <th>Titolo</th>
                <th>Autore</th>
                <th>Anno</th>
                <th>Azioni</th>
            </tr>
        </thead>
        <tbody id="bookTable">
            <!-- Le righe della tabella saranno generate dinamicamente -->
        </tbody>
    </table>

    <script>
        // Array per memorizzare i dati dei libri
        let books = [];
```

```
// Funzione per aggiungere un nuovo libro
function addBook() {
    // Ottiene i valori dai campi di input
    let title = document.getElementById("title").value;
    let author = document.getElementById("author").value;
    let year = document.getElementById("year").value;

    // Crea un oggetto per il nuovo libro e lo aggiunge alla matrice
    let book = { title, author, year };
    books.push(book);

    // Aggiorna la tabella e resetta i campi
    updateTable();
    document.getElementById("bookForm").reset();
}

// Funzione per aggiornare la tabella con i dati dell'array
function updateTable() {
    let table = document.getElementById("bookTable");
    table.innerHTML = "";  // Pulisce la tabella

    // Genera una nuova riga per ogni libro nella matrice
    books.forEach((book, index) => {
        let row = table.insertRow();
        row.insertCell(0).innerText = book.title;
        row.insertCell(1).innerText = book.author;
        row.insertCell(2).innerText = book.year;

        // Crea i pulsanti di modifica ed elimina
        let actionsCell = row.insertCell(3);
        actionsCell.innerHTML = `
            <button onclick="editBook(${index})">Modifica</button>
            <button onclick="deleteBook(${index})">Elimina</button>
        `;
    });
}

// Funzione per eliminare un libro
function deleteBook(index) {
    // Rimuove il libro dall'array usando l'indice
    books.splice(index, 1);

    // Aggiorna la tabella
    updateTable();
}

// Funzione per modificare un libro
function editBook(index) {
    // Ottiene i dati del libro selezionato
    let book = books[index];

    // Pre-popola i campi di input con i dati esistenti
    document.getElementById("title").value = book.title;
    document.getElementById("author").value = book.author;
    document.getElementById("year").value = book.year;

    // Aggiorna il pulsante per confermare le modifiche
    document.getElementById("bookForm").onsubmit = function(event) {
        event.preventDefault();  // Evita il comportamento predefinito
        updateBook(index);
    };
}
```

```
        // Funzione per confermare le modifiche di un libro
        function updateBook(index) {
            // Aggiorna i dati del libro con i valori nei campi di input
            books[index] = {
                title: document.getElementById("title").value,
                author: document.getElementById("author").value,
                year: document.getElementById("year").value
            };

            // Resetta il form e aggiorna la tabella
            document.getElementById("bookForm").reset();
            document.getElementById("bookForm").onsubmit = function(event) {
                event.preventDefault();
                addBook();
            };
            updateTable();
        }
    </script>
</body>
</html>
```

Spiegazione Dettagliata (CRUD di Base)

1. **Definizione Array `books`**:
   - Un array vuoto `books` memorizza i libri come oggetti `{ title, author, year }`.
2. **Funzione `addBook()`**:
   - Ottiene i valori dei campi di input.
   - Crea un oggetto libro e lo aggiunge all'array `books`.
   - Chiama `updateTable()` per visualizzare i dati nella tabella.
   - Resetta il form.
3. **Funzione `updateTable()`**:
   - Pulisce la tabella (`table.innerHTML = "";`).
   - Crea una nuova riga e celle con i dati per ogni oggetto `book`.
   - Aggiunge i pulsanti di modifica (`editBook(index)`) e di eliminazione (`deleteBook(index)`).
4. **Funzione `deleteBook(index)`**:
   - Elimina il libro dall'array `books` usando `splice()`.
5. **Funzione `editBook(index)`**:
   - Estrae i dati del libro da `books`.
   - Popola i campi del form e modifica il pulsante del form per confermare le modifiche con `updateBook(index)`.
6. **Funzione `updateBook(index)`**:
   - Aggiorna l'oggetto libro nell'array `books`.
   - Resetta il form e la tabella per visualizzare i dati aggiornati.

Esercizio 2: Gestione Inventario (CRUD con Filtri)

Descrizione: Realizza una pagina HTML per gestire un inventario con Nome Prodotto, Quantità e Prezzo. L'utente può inserire, modificare e cancellare prodotti e visualizzarli in una tabella. Aggiungi una funzionalità di ricerca per filtrare i prodotti visualizzati.

Esercizio 3: Gestione Prenotazioni (CRUD con Storage Locale e Ordinamento)

Descrizione: Una pagina HTML per gestire le prenotazioni (Nome, Data, Numero di Ospiti). L'utente può aggiungere, modificare e cancellare prenotazioni, che saranno salvate in `localStorage` per persistere i dati tra le sessioni. Inoltre, la tabella può essere ordinata per Data.

Esercizio 2: Gestione Inventario con Filtro

Descrizione del Problema

L'obiettivo è creare una pagina HTML per gestire l'inventario di prodotti, ognuno con un Nome, una Quantità e un Prezzo. L'utente può:

1. Aggiungere prodotti.
2. Modificare i dettagli di prodotti esistenti.
3. Eliminare prodotti.
4. Cercare prodotti tramite una barra di ricerca che filtra i risultati.

Codice Completo

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Gestione Inventario</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        form { margin-bottom: 20px; }
        input, button { padding: 8px; margin: 5px; }
        table { width: 100%; border-collapse: collapse; margin-top: 20px; }
        th, td { border: 1px solid #ddd; padding: 10px; text-align: left; }
        th { background-color: #f4f4f4; }
    </style>
</head>
<body>
    <h2>Gestione Inventario</h2>

    <!-- Campo di ricerca -->
    <input type="text" id="searchBar" placeholder="Cerca prodotto"
oninput="filterInventory()">

    <!-- Form di input per aggiungere un prodotto -->
    <form id="productForm">
        <input type="text" id="productName" placeholder="Nome Prodotto"
required>
        <input type="number" id="productQuantity" placeholder="Quantità"
required>
        <input type="number" id="productPrice" placeholder="Prezzo (€)"
required>
```

```
    <button type="button" onclick="addProduct()">Aggiungi Prodotto</button>
</form>

<!-- Tabella per visualizzare i prodotti -->
<table>
    <thead>
        <tr>
            <th>Nome</th>
            <th>Quantità</th>
            <th>Prezzo (€)</th>
            <th>Azioni</th>
        </tr>
    </thead>
    <tbody id="productTable">
        <!-- Le righe saranno generate dinamicamente -->
    </tbody>
</table>

<script>
    // Array per memorizzare i dati dei prodotti
    let products = [];

    // Funzione per aggiungere un nuovo prodotto
    function addProduct() {
        // Ottiene i valori dai campi di input
        let name = document.getElementById("productName").value;
        let quantity = document.getElementById("productQuantity").value;
        let price = document.getElementById("productPrice").value;

        // Crea un oggetto per il nuovo prodotto e lo aggiunge all'array
        let product = { name, quantity, price };
        products.push(product);

        // Aggiorna la tabella e resetta il form
        updateTable();
        document.getElementById("productForm").reset();
    }

    // Funzione per aggiornare la tabella
    function updateTable() {
        let table = document.getElementById("productTable");
        table.innerHTML = "";  // Pulisce la tabella

        // Genera una nuova riga per ogni prodotto
        products.forEach((product, index) => {
            let row = table.insertRow();
            row.insertCell(0).innerText = product.name;
            row.insertCell(1).innerText = product.quantity;
            row.insertCell(2).innerText = product.price;

            // Crea pulsanti di modifica ed elimina
            let actionsCell = row.insertCell(3);
            actionsCell.innerHTML = `
                <button onclick="editProduct(${index})">Modifica</button>
                <button onclick="deleteProduct(${index})">Elimina</button>
            `;
        });
    }

    // Funzione per eliminare un prodotto
    function deleteProduct(index) {
        // Rimuove il prodotto dall'array
        products.splice(index, 1);
```

```
            // Aggiorna la tabella
            updateTable();
        }

        // Funzione per modificare un prodotto
        function editProduct(index) {
            // Ottiene i dati del prodotto selezionato
            let product = products[index];

            // Pre-popola i campi di input con i dati esistenti
            document.getElementById("productName").value = product.name;
            document.getElementById("productQuantity").value = product.quantity;
            document.getElementById("productPrice").value = product.price;

            // Cambia il form per aggiornare invece di aggiungere un nuovo
prodotto
            document.getElementById("productForm").onsubmit = function(event) {
                event.preventDefault();
                updateProduct(index);
            };
        }

        // Funzione per confermare la modifica di un prodotto
        function updateProduct(index) {
            // Aggiorna l'oggetto prodotto con i nuovi valori dai campi di input
            products[index] = {
                name: document.getElementById("productName").value,
                quantity: document.getElementById("productQuantity").value,
                price: document.getElementById("productPrice").value
            };

            // Resetta il form e aggiorna la tabella
            document.getElementById("productForm").reset();
            document.getElementById("productForm").onsubmit = function(event) {
                event.preventDefault();
                addProduct();
            };
            updateTable();
        }

        // Funzione per filtrare i prodotti in base alla ricerca
        function filterInventory() {
            let query =
document.getElementById("searchBar").value.toLowerCase();
            let table = document.getElementById("productTable");
            table.innerHTML = "";  // Pulisce la tabella

            // Filtra i prodotti e crea le righe solo per quelli che
corrispondono alla ricerca
            products.filter(product =>
product.name.toLowerCase().includes(query)).forEach((product, index) => {
                let row = table.insertRow();
                row.insertCell(0).innerText = product.name;
                row.insertCell(1).innerText = product.quantity;
                row.insertCell(2).innerText = product.price;

                // Crea pulsanti di modifica ed elimina
                let actionsCell = row.insertCell(3);
                actionsCell.innerHTML = `
                    <button onclick="editProduct(${index})">Modifica</button>
                    <button onclick="deleteProduct(${index})">Elimina</button>
                `;
```

```
            });
        }
    </script>
</body>
</html>
```

Spiegazione del Codice JavaScript

1. **Array `products`**:
   - Array vuoto `products` in cui saranno salvati i prodotti.
2. **Funzione `addProduct()`**:
   - Raccoglie i valori da `productName`, `productQuantity`, e `productPrice`.
   - Crea un oggetto `product` e lo inserisce in `products`.
   - Chiama `updateTable()` per visualizzare i nuovi dati.
3. **Funzione `updateTable()`**:
   - Pulisce la tabella e crea una nuova riga per ciascun prodotto.
   - Aggiunge due pulsanti per modificare (`editProduct(index)`) o eliminare (`deleteProduct(index)`).
4. **Funzione `deleteProduct(index)`**:
   - Rimuove il prodotto dall'array `products` usando `splice`.
   - Chiama `updateTable()` per aggiornare la tabella.
5. **Funzione `editProduct(index)`**:
   - Pre-compila i campi del form con i valori del prodotto.
   - Modifica l'azione del form per aggiornare il prodotto con `updateProduct(index)`.
6. **Funzione `updateProduct(index)`**:
   - Aggiorna i valori del prodotto nell'array `products`.
   - Resetta il form per ritornare alla modalità di aggiunta.
7. **Funzione `filterInventory()`**:
   - Filtra l'array `products` in base alla ricerca e crea una riga per ogni elemento corrispondente.

Procedo con il **Esercizio 3: Gestione Prenotazioni con LocalStorage e Ordinamento**, includendo la descrizione del problema, il codice completo e la spiegazione dettagliata.

Esercizio 3: Gestione Prenotazioni (CRUD con LocalStorage e Ordinamento)

Descrizione del Problema

Questo esercizio consiste nella creazione di una pagina HTML per gestire un elenco di prenotazioni. Ogni prenotazione include:

- Nome del cliente
- Data della prenotazione
- Numero di ospiti

L’utente può aggiungere, modificare, cancellare e visualizzare le prenotazioni. Inoltre:

1. I dati devono essere memorizzati in `localStorage`, così da persistere anche dopo il riavvio della pagina.
2. È presente un'opzione per ordinare le prenotazioni per data.

Codice Completo

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Gestione Prenotazioni</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        form { margin-bottom: 20px; }
        input, button { padding: 8px; margin: 5px; }
        table { width: 100%; border-collapse: collapse; margin-top: 20px; }
        th, td { border: 1px solid #ddd; padding: 10px; text-align: left; }
        th { background-color: #f4f4f4; }
        th.sortable:hover { cursor: pointer; color: blue; }
    </style>
</head>
<body>
    <h2>Gestione Prenotazioni</h2>

    <!-- Form di input per aggiungere una prenotazione -->
    <form id="reservationForm">
        <input type="text" id="customerName" placeholder="Nome Cliente"
required>
        <input type="date" id="reservationDate" required>
        <input type="number" id="guestCount" placeholder="Numero di Ospiti"
required>
        <button type="button" onclick="addReservation()">Aggiungi
Prenotazione</button>
    </form>

    <!-- Tabella per visualizzare le prenotazioni -->
    <table>
        <thead>
            <tr>
                <th>Nome</th>
                <th class="sortable" onclick="sortReservations()">Data</th>
                <th>Numero di Ospiti</th>
                <th>Azioni</th>
            </tr>
        </thead>
        <tbody id="reservationTable">
            <!-- Le righe saranno generate dinamicamente -->
        </tbody>
    </table>

    <script>
        // Array per memorizzare le prenotazioni
        let reservations = JSON.parse(localStorage.getItem("reservations")) ||
[];

        // Carica le prenotazioni salvate alla pagina
        updateTable();
```

```
        // Funzione per aggiungere una nuova prenotazione
        function addReservation() {
            let name = document.getElementById("customerName").value;
            let date = document.getElementById("reservationDate").value;
            let guests = document.getElementById("guestCount").value;

            // Crea un oggetto prenotazione e lo aggiunge all'array
            let reservation = { name, date, guests };
            reservations.push(reservation);

            // Salva in localStorage e aggiorna la tabella
            localStorage.setItem("reservations", JSON.stringify(reservations));
            updateTable();
            document.getElementById("reservationForm").reset();
        }

        // Funzione per aggiornare la tabella
        function updateTable() {
            let table = document.getElementById("reservationTable");
            table.innerHTML = "";  // Pulisce la tabella

            reservations.forEach((reservation, index) => {
                let row = table.insertRow();
                row.insertCell(0).innerText = reservation.name;
                row.insertCell(1).innerText = reservation.date;
                row.insertCell(2).innerText = reservation.guests;

                // Crea pulsanti di modifica ed elimina
                let actionsCell = row.insertCell(3);
                actionsCell.innerHTML = `
                    <button
onclick="editReservation(${index})">Modifica</button>
                    <button
onclick="deleteReservation(${index})">Elimina</button>
                `;
            });
        }

        // Funzione per eliminare una prenotazione
        function deleteReservation(index) {
            reservations.splice(index, 1);
            localStorage.setItem("reservations", JSON.stringify(reservations));
            updateTable();
        }

        // Funzione per modificare una prenotazione
        function editReservation(index) {
            let reservation = reservations[index];
            document.getElementById("customerName").value = reservation.name;
            document.getElementById("reservationDate").value = reservation.date;
            document.getElementById("guestCount").value = reservation.guests;

            document.getElementById("reservationForm").onsubmit =
function(event) {
                event.preventDefault();
                updateReservation(index);
            };
        }

        // Funzione per confermare la modifica di una prenotazione
        function updateReservation(index) {
            reservations[index] = {
                name: document.getElementById("customerName").value,
```

```
                date: document.getElementById("reservationDate").value,
                guests: document.getElementById("guestCount").value
            };
            localStorage.setItem("reservations", JSON.stringify(reservations));
            document.getElementById("reservationForm").reset();
            document.getElementById("reservationForm").onsubmit =
function(event) {
                event.preventDefault();
                addReservation();
            };
            updateTable();
        }

        // Funzione per ordinare le prenotazioni per data
        function sortReservations() {
            reservations.sort((a, b) => new Date(a.date) - new Date(b.date));
            localStorage.setItem("reservations", JSON.stringify(reservations));
            updateTable();
        }
    </script>
</body>
</html>
```

Spiegazione Dettagliata (Riga per Riga)

1. **Inizializzazione Array `reservations`**:
   - `JSON.parse(localStorage.getItem("reservations")) || []` carica le prenotazioni salvate in `localStorage` oppure un array vuoto se non esistono prenotazioni.
2. **Funzione `addReservation()`**:
   - Raccoglie i valori dei campi `customerName`, `reservationDate`, e `guestCount`.
   - Crea un oggetto `reservation` con questi valori e lo aggiunge all'array `reservations`.
   - Salva `reservations` in `localStorage` (con `setItem` e `JSON.stringify`) e aggiorna la tabella con `updateTable()`.
3. **Funzione `updateTable()`**:
   - Cancella le righe esistenti e genera una nuova riga per ogni prenotazione.
   - Ogni riga include due pulsanti:
     - **Modifica** (`editReservation(index)`) e **Elimina** (`deleteReservation(index)`).
4. **Funzione `deleteReservation(index)`**:
   - Usa `splice(index, 1)` per rimuovere la prenotazione dall'array `reservations`.
   - Aggiorna `localStorage` e chiama `updateTable()`.
5. **Funzione `editReservation(index)`**:
   - Estrae l'oggetto prenotazione da `reservations`.
   - Riempie i campi del form con i dati esistenti.
   - Cambia il pulsante del form per aggiornare la prenotazione usando `updateReservation(index)`.
6. **Funzione `updateReservation(index)`**:
   - Aggiorna `reservations[index]` con i valori nei campi di input.
   - Salva `reservations` in `localStorage`, resetta il form e ritorna alla funzione `addReservation`.
7. **Funzione `sortReservations()`**:

- Usa `sort()` per ordinare `reservations` per data:

  ```
  javascript
  Copia codice
  reservations.sort((a, b) => new Date(a.date) - new Date(b.date));
  ```

- Aggiorna `localStorage` e la tabella.

Con queste funzionalità, il progetto permette una gestione efficace delle prenotazioni, mantenendo i dati sincronizzati in `localStorage` e aggiungendo la possibilità di ordinamento per data.