Esercitazione ES6: sviluppare un'applicazione di tipo BLOG che ottiene dati JSON da diverse API

Titolo: Esercitazione ES6 - Applicazione Blog

Obiettivo: In questa esercitazione, creerete un'applicazione di blog che otterrà dati JSON da tre diverse API: post, commenti e utenti da https://jsonplaceholder.typicode.com/. Useremo ES6, Promises, async/await e l'API fetch per gestire le richieste e visualizzare i dati.

Istruzioni:

1. Creare un file HTML e inserire il seguente codice HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Blog App</title>
</head>
<body>
  <h1>Lista di Post</h1>
  <ul id="post-list"></ul>

  <script>
    // Inserire il codice JavaScript qui
  </script>
</body>
</html>
```

2. All'interno della sezione JavaScript (`<script>`), creare tre funzioni asincrone: `fetchPosts`, `fetchUsers`, e `fetchComments`. Queste funzioni dovrebbero utilizzare `fetch` per ottenere i dati JSON dalle rispettive API (posts, users e comments) e restituire i dati.

3. Creare una funzione asincrona `renderBlog` che chiama le tre funzioni di cui sopra per ottenere i dati dei post, utenti e commenti.

4. Utilizzare le informazioni ottenute per creare una lista di post nel documento HTML. Per ciascun post, visualizzare il titolo, il corpo del post, il nome dell'autore e il numero di commenti associati.

5. Testare l'applicazione aprendo il file HTML in un browser.

Note:

- Assicurarsi che il vostro computer abbia accesso a Internet perché l'applicazione farà richieste alle API online.

- È possibile utilizzare le funzioni `console.log` per esaminare i dati ottenuti dalle API durante lo sviluppo.
- Prestare attenzione agli errori durante le richieste e gestirli correttamente nel codice.

Ecco un esempio di esercitazione in cui utilizzeremo ES6 per creare un'applicazione di blog che ottiene dati JSON da tre diverse API (posts, comments e users) da https://jsonplaceholder.typicode.com/. Useremo le Promises, `async`/`await`, e `fetch` per gestire le richieste e visualizzare i dati. Assicurati di avere una connessione Internet attiva per eseguire questo esercizio.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Blog App</title>
</head>
<body>
  <h1>Lista di Post</h1>
  <ul id="post-list"></ul>

  <script>
    async function fetchPosts() {
      try {
        const response = await
fetch('https://jsonplaceholder.typicode.com/posts');
        if (!response.ok) {
          throw new Error('Errore nella richiesta dei post');
        }
        const posts = await response.json();
        return posts;
      } catch (error) {
        console.error(error);
      }
    }

    async function fetchUsers() {
      try {
        const response = await
fetch('https://jsonplaceholder.typicode.com/users');
        if (!response.ok) {
          throw new Error('Errore nella richiesta degli utenti');
        }
        const users = await response.json();
        return users;
      } catch (error) {
        console.error(error);
      }
    }

    async function fetchComments() {
      try {
```

```
        const response = await
fetch('https://jsonplaceholder.typicode.com/comments');
        if (!response.ok) {
          throw new Error('Errore nella richiesta dei commenti');
        }
        const comments = await response.json();
        return comments;
      } catch (error) {
        console.error(error);
      }
    }

    async function renderBlog() {
      const posts = await fetchPosts();
      const users = await fetchUsers();
      const comments = await fetchComments();

      const postList = document.getElementById('post-list');
      posts.forEach((post) => {
        const user = users.find((u) => u.id === post.userId);
        const postComments = comments.filter((comment) => comment.postId
=== post.id);

        const listItem = document.createElement('li');
        listItem.innerHTML = `
          <h2>${post.title}</h2>
          <p>${post.body}</p>
          <p>Autore: ${user.name}</p>
          <p>Commenti: ${postComments.length}</p>
        `;
        postList.appendChild(listItem);
      });
    }

    renderBlog();
  </script>
</body>
</html>
```

Questo esercizio crea una pagina web che visualizza una lista di post ottenuti dalla risorsa `posts` e visualizza l'autore di ciascun post e il numero di commenti associati. Abbiamo utilizzato `async`/`await` per rendere le chiamate API asincrone e `fetch` per ottenere i dati. Inoltre, abbiamo gestito gli errori durante le richieste.