ステップ4: タスク完了状態の保存と復元（完了状態の永続化）（穴埋め版）

導入・問題提起

前回のステップ3で「タスクの追加と復元」機能を実装しました。 しかし、現在はタスクの完了状態（チェックボックス）を切り替えても、その状態がlocalStorageに保存されません。

今回は、タスクの完了状態を切り替えた時も自動的にlocalStorageに保存し、ページを再読み込みしても完了状態が維持される機能を実装します。

現在の問題:

- タスクを追加 → ✅ localStorage保存される
- ページ再読み込み → ✅ タスク一覧が復元される
- タスクの完了切り替え → ❌ **localStorage保存されない**
- ページ再読み込み → ❌ **完了状態がリセットされる**

実装後の状態:

- タスクの完了状態を切り替え → ✅ localStorage保存される
- ページ再読み込み → ✅ 完了状態も含めて完全復元される

このステップの目標

- タスクの完了/未完了状態がlocalStorageに正しく保存・復元されることを理解する
- オブジェクトのプロパティが変更されても自動保存されない理由を学ぶ
- イベントハンドラー内での手動保存の必要性を理解する
- 完全なデータ永続化の仕組みを習得する

考えてみよう

手順1: 現在のtoggleTask関数の確認

まず、現在のtoggleTask関数がどのように動作しているかを確認します。

```
// 現在のtoggleTask関数（localStorage保存なし）
function toggleTask(id) {
    // 指定されたIDのタスクを見つけて完了状態を切り替え
    tasks.forEach(function (task) {
        if (task.id === id) {
            task.completed = !task.completed;
            console.log(`🔄 タスク「${task.text}」の完了状態を切り替えました`);
        }
    });

    // 画面表示を更新
    renderTasks();
    updateProgress();
}
```

手順2: localStorage保存機能の追加

toggleTask関数にlocalStorage保存機能を追加します。

```
function toggleTask(id) {
    // 指定されたIDのタスクを見つけて完了状態を切り替え
    tasks.forEach(function (task) {
        if (task.id === id) {
            task.completed = !task.completed;
            console.log(`🔄 タスク「${task.text}」の完了状態を${task.completed ? '完了' : '未完了'}に切り替えました`);
        }
    });

    // 【穴埋め①】完了状態の変更をlocalStorageに保存する関数を呼び出す（20文字）
    ____________________;

    // 画面表示を更新
    renderTasks();
    updateProgress();

    console.log("💾 完了状態の変更をlocalStorageに保存しました");
}
```

手順3: 動作確認

実装完了後、以下の手順で動作確認を行います。

1. **タスクを追加**（例：「買い物」）

2. **チェックボックスをクリック**して完了状態にする
3. **DevToolsのConsole**で保存メッセージを確認
4. **ページを再読み込み**
5. **結果確認**：タスクが完了状態で復元されることを確認

✨ 最小限ヒント

- ステップ2で作成した`saveTasksToStorage()`関数を再利用します。
- データの状態（この場合はタスクの`completed`プロパティ）が変更された直後に、`saveTasksToStorage()`を呼び出してlocalStorageに最新の状態を保存する必要があります。
- `toggleTask`関数内で、`task.completed = !task.completed;` の行のすぐ後が適切な呼び出し場所です。
- ステップ2_タスク追加時の自動保存機能_穴埋め版.mdの`addTask`関数での`saveTasksToStorage()`の使われ方も参考にしてください。

解答例

```
function toggleTask(id) {
    // 指定されたIDのタスクを見つけて完了状態を切り替え
    tasks.forEach(function (task) {
        if (task.id === id) {
            task.completed = !task.completed;
            console.log(`🔄 タスク「${task.text}」の完了状態を${task.completed ? '完了' : '未完了'}に切り替えました`);
        }
    });

    // 完了状態の変更をlocalStorageに保存
    saveTasksToStorage(); // この行を追加

    // 画面表示を更新
    renderTasks();
    updateProgress();

    console.log("💾 完了状態の変更をlocalStorageに保存しました");
}
```

ポイント解説

① 完全な永続化への一歩

このステップの修正により、タスクの「完了状態」もlocalStorageに保存・復元されるようになり、アプリケーションのデータ永続性がさらに向上しました。ユーザーがタスクのチェック状態を変更した後、ページを再読み込みしてもその状態が保持されます。

② オブジェクトプロパティ変更時の明示的な保存

JavaScriptの配列やオブジェクトの内容（例えば、タスクオブジェクトの `completed` プロパティ）を変更した場合、その変更をlocalStorageに反映させるためには、再度 `JSON.stringify()` で全体のデータを文字列化し、`localStorage.setItem()` で保存し直す必要があります。 `saveTasksToStorage()` 関数がこの役割を担っています。

③ ユーザー操作とデータ同期の原則

ユーザーが行う操作（タスク追加、完了状態の切り替え、タスク削除など）によってアプリケーションのデータが変更される場合、その都度 `saveTasksToStorage()` のような関数を呼び出して、変更後の最新データをlocalStorageと同期させることが、データの一貫性を保つための基本です。

補足資料への誘導

- **localStorageの挙動と注意点**： `補足資料/localStorage_API詳解.md`
- **オブジェクトと配列のJSON変換**： `補足資料/JSON操作詳解.md`
- **イベント処理とデータ更新**： `補足資料/イベント駆動プログラミング基礎.md` （もしあれば）

動作確認チェックリスト

実装が完了したら、以下の手順で動作を確認しましょう。

1. **タスクをいくつか追加**します（例: 「牛乳を買う」「レポートを書く」「運動する」）。
2. **いずれかのタスクのチェックボックスをクリック**して、完了状態にしてみましょう。
    - ☐ Consoleに `🔄 タスク「（タスク名）」の完了状態を完了に切り替えました` と表示されるか。
    - ☐ Consoleに `💾 完了状態の変更をlocalStorageに保存しました` と表示されるか。
    - ☐ ApplicationタブのLocal Storageで、該当タスクの `completed` プロパティが `true` に更新されているか。
3. **別のタスクのチェックボックスを何度かクリック**して、完了・未完了を切り替えてみましょう。
    - ☐ その都度、Consoleに適切なメッセージが表示され、Local Storageの内容も更新されるか。
4. **ページを再読み込み（F5キー）**します。

- ページ再読み込み後、各タスクの完了状態（チェックボックスの状態）が、再読み込み前の最後の状態どおりに復元されているか。
- 進捗バーが、復元されたタスクの完了状態を正しく反映しているか。

5. **全てのタスクを完了状態に**してみてください。
   - 進捗バーが100%になり、「全てのタスク完了！」と表示されるか。
   - この状態でページを再読み込みしても、全てのタスクが完了状態で復元され、進捗バーも100%のままか。

次のステップへの案内

ステップ4でタスク完了状態の永続化ができました。次のステップ5では、タスク削除操作もlocalStorageに正しく反映させ、CRUD操作（作成・読み取り・更新・削除）の全てでlocalStorageとの同期を完成させます。