ステップ3: ページ読み込み時のタスク復元機能（JSON解析 & 初期化処理）（穴埋め版）

導入・問題提起

前回のステップ2で「タスクを追加したときにlocalStorageに保存する」機能を実装しました。 しかし、ページを再読み込みしても保存したタスクが表示されません。これは、保存はできているものの、ページを開いたときにそのデータを読み込んで画面に表示する仕組みがまだないためです。

今回は、ページが読み込まれた時にlocalStorageから保存されたタスクを取得して、自動的に画面に復元する機能を実装します。

Before（現在の状態）：

- タスクを追加すると自動的にlocalStorageに保存される
- しかし、ページを再読み込みするとタスク一覧は空になってしまう

このステップの目標

- 保存されたタスクデータをページ読み込み時に自動復元できるようになる
- `JSON.parse()` を使ってJSON文字列をJavaScriptオブジェクトに変換できる
- localStorageの値が `null` の場合の適切な処理ができる
- `DOMContentLoaded` イベントを活用したページ初期化処理を理解する

After（実装後の状態）：

- ページを読み込むと、前回保存したタスクが自動的に画面に表示される
- localStorage内のタスクデータが完全に復元される

考えてみよう

手順1: localStorage読み込み関数の作成

まず、localStorageからタスクデータを読み込む関数を作成します。

```
function loadTasksFromStorage() {
    try {
        // 【穴埋め①】localStorageから'todoTasks'キーの値を取得する
        const savedTasksJson = localStorage.getItem('todoTasks');

        // 【穴埋め②】取得した値がnullかどうかをチェックする条件文
        if (savedTasksJson === null) {
            console.log("📄 保存されたタスクデータがありません");
            return; // データがなければここで処理を終了
        }

        // 【穴埋め③】JSON文字列をJavaScriptオブジェクトに変換する
        const savedTasks = JSON.parse(savedTasksJson);

        // グローバルなtasks配列を復元されたデータで置き換え
        tasks = savedTasks; // tasks配列を直接更新

        console.log("📂 localStorageからタスクを復元しました:", savedTasks);

    } catch (error) {
        console.error("❌ localStorage読み込みエラー:", error);
        // エラー発生時は空のタスクリストとして扱うなどのフォールバック処理も考えられる
        tasks = []; // 安全のためtasksを空にする
        console.log("🔄 空のタスクリストで開始します");
    }
}
```

手順2: ページ読み込み時の初期化処理

DOMContentLoadedイベントにタスク復元処理を追加します。

```
document.addEventListener('DOMContentLoaded', function() {
    const taskInput = document.getElementById('taskInput');
    const addButton = document.getElementById('addButton');

    // 【穴埋め④】localStorageからタスクを読み込む関数を呼び出す
    loadTasksFromStorage(); // localStorageからタスクを読み込む

    // Enterキーでタスク追加
    taskInput.addEventListener('keypress', function(e) {
        if (e.key === 'Enter') {
            addTask();
        }
    });
```

```
    // ボタンクリックでタスク追加
    addButton.addEventListener('click', addTask);

    // 【穴埋め⑤】復元したタスクを画面に表示する関数を呼び出す
    renderTasks(); // 読み込んだタスクを画面に表示する

    // 【穴埋め⑥】進捗バーを更新する関数を呼び出す
    updateProgress(); // 進捗バーを更新する
});
```

✨ 最小限ヒント

- localStorage.getItem()で保存されたデータを取得
- 取得した値が`null`の場合は、データが存在しない状態
- JSON.parse()で文字列をJavaScriptオブジェクトに復元
- DOMContentLoadedイベントでページ読み込み完了時に実行
- ステップ2: タスク追加時の自動保存機能のSTORAGE_KEY（`'todoTasks'`）を使用

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、localStorageにタスクデータが保存されることをブラウザのDevTools（開発者ツール）で確認します（Application > Local Storageで `todoTasks` キーの値を確認）。
2. **ページを再読み込み**します。
3. **確認事項:**
    - ☐ ページ再読み込み後、以前追加したタスクがすべて画面に表示されているか。
    - ☐ タスクの内容（テキスト）や未完了状態が正しく復元されているか。
    - ☐ DevToolsのConsoleに”📂 localStorageからタスクを復元しました:”というメッセージと復元されたタスクデータが表示されるか（データがある場合）。
    - ☐ （テストケース）localStorageに `'todoTasks'` のデータが存在しない状態でページを開いた場合（DevToolsで手動削除するなどしてテスト）、Consoleに”📄 保存されたタスクデータがありません”と表示され、エラーなくページが表示されるか。
    - ☐ （テストケース）localStorageの `'todoTasks'` に不正なJSON文字列（例: `{"text": "test}` のような壊れたJSON）を保存して（DevToolsで手動編集してテスト）、ページを再読み込みした場合、Consoleに”❌ localStorage読み込みエラー:”というメッセージとエラー詳細が表示され、”🔄 空のタスクリストで開始します”と表示されるか。

解答例

手順1: localStorage読み込み関数の作成（`loadTasksFromStorage`）

```
function loadTasksFromStorage() {
    try {
        // 【穴埋め①】localStorageから'todoTasks'キーの値を取得する
        const savedTasksJson = localStorage.getItem('todoTasks');

        // 【穴埋め②】取得した値がnullかどうかをチェックする条件文
        if (savedTasksJson === null) {
            console.log("📄 保存されたタスクデータがありません");
            return; // データがなければここで処理を終了
        }

        // 【穴埋め③】JSON文字列をJavaScriptオブジェクトに変換する
        const savedTasks = JSON.parse(savedTasksJson);

        // グローバルなtasks配列を復元されたデータで置き換え
        tasks = savedTasks; // tasks配列を直接更新

        console.log("📂 localStorageからタスクを復元しました:", savedTasks);

    } catch (error) {
        console.error("❌ localStorage読み込みエラー:", error);
        // エラー発生時は空のタスクリストとして扱うなどのフォールバック処理も考えられる
        tasks = []; // 安全のためtasksを空にする
        console.log("🔄 空のタスクリストで開始します");
    }
}
```

手順2: ページ読み込み時の初期化処理（`DOMContentLoaded`）

```
document.addEventListener('DOMContentLoaded', function() {
    const taskInput = document.getElementById('taskInput');
    const addButton = document.getElementById('addButton');

    // 【穴埋め④】localStorageからタスクを読み込む関数を呼び出す
    loadTasksFromStorage(); // localStorageからタスクを読み込む

    // Enterキーでタスク追加
    taskInput.addEventListener('keypress', function(e) {
        if (e.key === 'Enter') {
```

```
            addTask();
        }
    });

    // ボタンクリックでタスク追加
    addButton.addEventListener('click', addTask);

    // 【穴埋め⑤】復元したタスクを画面に表示する関数を呼び出す
    renderTasks(); // 読み込んだタスクを画面に表示する

    // 【穴埋め⑥】進捗バーを更新する関数を呼び出す
    updateProgress(); // 進捗バーを更新する
});
```

ポイント解説

① データ永続化の完成サイクル

今回でデータ保存（ステップ2）と復元（ステップ3）が完成し、基本的な永続化が実現できました。

```
// 保存: JavaScript → JSON → localStorage
const tasksJson = JSON.stringify(tasks);
localStorage.setItem('key', tasksJson);

// 復元: localStorage → JSON → JavaScript
const savedJson = localStorage.getItem('key');
const savedTasks = JSON.parse(savedJson);
```

② エラーハンドリングの重要性

localStorage操作では常にエラーの可能性があります：

- 容量制限によるQuotaExceededError
- 不正JSON文字列によるSyntaxError
- try-catch文で安全な処理を実現

③ ページ初期化パターン

```
document.addEventListener('DOMContentLoaded', function() {
    // 1. データ復元
    loadTasksFromStorage();
    // 2. 画面更新
```

```
    renderTasks();
    updateProgress();
    // 3. イベント設定
    setupEventListeners();
});
```

次のステップでは、タスクの完了状態も保存・復元できるようにします。

補足資料への誘導

- **JSON操作の詳細**： 補足資料/JSON操作詳解.md
- **localStorage API**： 補足資料/localStorage_API詳解.md
- **データ構造設計**： 補足資料/データ構造設計.md