ステップ2: タスク追加時の自動保存 機能 (JSON保存 & エラーハンドリン グ) (穴埋め版)

導入・問題提起

ステップ1ではlocalStorageの基本操作を学びましたが、まだ実際のタスク管理 アプリにはその機能が組み込まれていません。そのため、タスクを追加してもブラウザを更新すると消えてしまいます。

現在のアプリの問題を体験

現在のベースコードを使って以下を確認してください:

- 1. タスクを2-3個追加する
- 2. ブラウザを更新 (F5キー)
- 3. → 追加したタスクが全て消えることを確認

なぜこの問題が起きるのか? 現在のアプリは、タスクをJavaScriptの配列 (メモリ) にのみ保存しています。 ブラウザを更新すると、メモリの内容がリセットされるため、データが失われます。

Before(現在の状況)

- タスクを追加しても、ブラウザを更新すると消えてしまう
- ステップ1でlocalStorageの基本操作は学んだが、実際のアプリに活用できていない

このステップの目標

このステップを終えると、タスク管理アプリに以下の機能が実装され、動作するようになります。

- タスクを追加すると 自動的にlocalStorageに保存 される機能を実装できる
- JSON.stringify() を使ってオブジェクト配列をlocalStorageに保存できる

• try-catch 文でlocalStorage操作のエラーハンドリングができる

期待する動作:

- タスク追加と同時にlocalStorageにJSON形式で保存される
- エラーが発生した場合は適切にハンドリングされる
- タスク「localStorage学習」を追加すると、コンソールに保存完了メッセージが表示され、ブラウザの開発者ツール (Application > Local Storage) でデータを確認できる。 (注意:このステップでは保存のみを扱うため、ブラウザを更新するとタスクはまだ画面から消えます。復元機能はステップ3で実装します。)

考えてみよう: saveTasksToStorage関数の実装

ベースコードの script.js に以下の機能を追加しましょう。

手順1: localStorageのキー名を定数で定義

```
// タスクデータ(配列形式) - まだlocalStorageからの読み込みなし
let tasks = [];

// ★ 追加: localStorageのキー名を定数として定義
const STORAGE_KEY = "_____";
```

指示: localStorageで使用するキー名を"taskList"として定義してくださ

い。

手順2: saveTasksToStorage関数の作成

```
// タスクを追加する関数 (まだlocalStorage保存なし) function addTask() {
    // ... 既存のコード ...

// ★ この位置に保存機能を追加します (手順3で実装)

// 画面に表示
    renderTasks();
    updateProgress();
}

// ★ 追加: タスクをlocalStorageに保存する関数
function saveTasksToStorage() {
```

```
// 配列をJSON文字列に変換してlocalStorageに保存
const tasksJson = JSON. _____(tasks);
localStorage. _____(STORAGE_KEY, tasksJson);
console.log("■ localStorage保存完了:", tasksJson);

_____(error) {
console.error("ズ localStorage保存エラー:", error);
alert("データの保存に失敗しました");
}
```

指示:

- 1. 1つ目の : エラーハンドリングの開始キーワード (3文字) を入力してください
- 2. JSON._____:オブジェクトをJSON文字列に変換するメソッド名(9文字)を入力してください
- 3. localStorage. : localStorageにデータを保存するメソッド名(7文字)を入力してください
- 4. 2つ目の : エラーが発生した時の処理を記述するキーワード(5文字)を入力してください

手順3: addTask関数に保存機能を追加

```
function addTask() {
 const taskText = taskInput.value.trim();
 if (taskText === "") {
   alert("タスクを入力してください");
   return;
 }
 // 新しいタスクオブジェクトを作成
 const newTask = {
   id: Date.now(),
   text: taskText,
   completed: false
 // タスク配列に追加
 tasks.push(newTask);
 // 入力フィールドをクリア
 taskInput.value = "";
 // ★ 追加: localStorageに自動保存
 ____();
 // 画面に表示
 renderTasks();
 updateProgress();
```

console.log("<mark>✓</mark> タスクを追加し、localStorageに保存しました:", newTask.text);

指示: にlocalStorageへの保存を実行する関数名(18文字)を 入力してください。

☆ 最小限ヒント

try-catch文について:

- localStorage操作は失敗する可能性があります(容量不足、プライベートモードなど)
- try-catch文でエラーをキャッチし、ユーザーに分かりやすいメッセージを表示

JSON, stringify()について:

- JavaScriptのオブジェクトや配列をJSON文字列に変換
- localStorageは文字列しか保存できないため、オブジェクトは変換が必要

定数STORAGE_KEYについて:

- 複数の場所で同じキー名を使用するため、定数化で管理を簡単に
- タイポによるバグを防止

動作確認チェックリスト

実装が完了したら、以下の手順で動作確認をしてください:

- 1. タスク入力フィールドに何か入力して「追加」ボタンをクリックします。
- 2. ブラウザの開発者ツール (F12キー) を開き、Consoleタブを確認 します。
 - □ localStorage保存完了: [...] のようなメッセージが表示されていること。
 - □ メッセージ内のJSON文字列が、追加したタスクを含む配列の形式になっていること。
- 3. 開発者ツールのApplicationタブ(またはStorageタブ)を開き、Local Storageを選択 します。
 - http://127.0.0.1:5500 (または同様のローカルサーバーアドレス)の下に、STORAGE_KEY で定義したキー(例: taskList)が存在すること。
 - □ そのキーの値が、Consoleタブで確認したJSON文字列と一致していること。

- 4. **さらにタスクをいくつか追加** し、上記2,3の手順でlocalStorageの内容が更新されることを確認します。
- 5. **(エラーハンドリングテスト)** もし可能であれば、ブラウザの機能でlocalStorageの保存容量を一時的に非常に小さく設定するか、書き込みをブロックするような拡張機能を使って、保存エラーを意図的に発生させてみてください。
 - □ **×** localStorage保存エラー: というメッセージとエラー詳細がConsoleに表示されること。
 - □ データの保存に失敗しました というアラートが表示されること。
- 6. **ブラウザを更新(F5キー)**します。
 - □ この時点では、タスクは画面から消えます(復元機能は次のステップで実装するため、これは期待される動作です)。
 - □ しかし、ApplicationタブのLocal Storageには、更新前に保存されたタスクデータが残っていることを確認してください。

解答例

▶ 解答例(完全なコード)

ポイント解説

JSON.stringify()の役割

- 目的: JavaScriptのオブジェクトを文字列形式に変換
- なぜ必要: localStorageは文字列しか保存できないため
- **変換例**: {id: 1, text: "学習"} → ("id":1,"text":"学習"}"

try-catch文によるエラーハンドリング

- localStorage操作が失敗する場合:
 - ブラウザの容量制限に達した時
 - プライベートブラウジングモード
 - ブラウザの設定でlocalStorageが無効化されている時
- ▶ <mark>対策</mark>: try-catch文でエラーをキャッチし、ユーザーに適切なフィードバックを提供

定数の活用

STORAGE KEY: 複数箇所で使用するキー名を定数化

・ メリット:

- タイポによるバグを防止
- 変更時の修正箇所を一箇所に集約
- コードの可読性向上

補足資料への誘導

• 「補足資料2: JSON操作詳解」で、JSON.stringify()とJSON.parse()の詳細な仕組みや注意点を段階的に学べます。

次のステップへの案内

ステップ2でタスクの 保存 機能を実装しました。次のステップ3では、ページ読み込み時に10ca1Storageからタスクを 復元 する機能を実装し、完全な永続化を実現します!