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）

導入・問題提起

ステップ1でlocalStorageの基本操作を学びましたが、現在のToDoアプリは**タスクを追加してもlocalStorageに保存されません**。つまり、ブラウザを更新すると今まで通りデータが消えてしまいます。

この問題を解決するため、**タスク追加時に自動的にlocalStorageへ保存する機能**を実装しましょう。

このステップの目標

Before（現在の状況）

タスク追加 → メモリのみに保存 → ブラウザ更新 → データ消失 😢

After（このステップ完了後）

タスク追加 → localStorage自動保存 → ブラウザ更新 → まだ表示は消える*
(*復元機能はステップ3で実装)

ただし：開発者ツールのLocal Storageには保存されている！ 🎉

具体的にできるようになること：

- タスク追加と同時にlocalStorageへ自動保存する機能を実装できる
- オブジェクト配列をJSON形式で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

現在のアプリの問題を確認しよう

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**（例：「勉強する」「買い物に行く」「メールを返信する」）
2. **F12キーで開発者ツールを開く → Application タブ → Local Storage を確認**
3. → まだ何もデータが保存されていないことを確認
4. **ブラウザを更新（F5キー）**
5. → 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。
> localStorage への保存処理を実装していないため、データの永続化ができていません。
>
> 詳細は → 補足資料：ブラウザストレージ比較

✅ 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で使用するキー名を定数として定義
const STORAGE_KEY = "taskList";
```

手順2：saveTasksToStorage関数の作成

```
// ★ 追加：タスクをlocalStorageに保存する関数
function saveTasksToStorage() {
  try {
    // 配列をJSON文字列に変換してlocalStorageに保存
```

```
    const tasksJson = JSON.stringify(tasks);
    localStorage.setItem(STORAGE_KEY, tasksJson);
    console.log("💾 localStorage保存完了:", tasksJson);
  } catch (error) {
    console.error("❌ localStorage保存エラー:", error);
    alert("データの保存に失敗しました");
  }
}
```

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
  saveTasksToStorage();

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

  console.log("✅ タスクを追加し、localStorageに保存しました:", newTask.text);
}
```

ポイント解説

try-catch文の基本構文:

```
try {
  // 失敗する可能性のある処理
} catch (error) {
  // エラーが発生した時の処理
}
```

JSON変換の基本:

- `JSON.stringify(オブジェクト)` → オブジェクトを文字列に変換
- `localStorage.setItem(キー, 値)` → localStorageに保存

localStorage保存の失敗例:

- ブラウザの容量制限（通常5MB）を超過
- プライベートブラウジングモード
- ブラウザ設定でlocalStorageが無効

> **詰まったら...** 補足資料: localStorage API詳解 の「詳細(3分)」セクション を参照してください。

補足資料への誘導

このステップの理解をさらに深めるために、以下の補足資料も参照してください。

- **localStorageの基本的な使い方**: 補足資料: localStorage API詳解
- **JSON形式への変換と注意点**: 補足資料: JSON操作詳解
- **エラー発生時の対処法**: 補足資料: try-catch詳解
- **ブラウザのストレージ種類と比較**: 補足資料: ブラウザストレージ比較

動作確認

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. コンソールでの確認

1. ブラウザの開発者ツール（F12）を開く
2. Consoleタブを選択
3. タスクを追加
4. → `💾 localStorage保存完了: [{"id":...}]` のようなメッセージが表示されるか確認

2. Application > Local Storageでの確認

1. 開発者ツールのApplicationタブを選択
2. 左側メニューのLocal Storage > http://localhost:5500 をクリック
3. キー "taskList" の値にJSON形式のタスクデータが保存されているか確認

3. ブラウザ更新での動作確認

1. タスクを2-3個追加
2. F5キーでブラウザを更新
3. → **まだタスクは消える**（これは正常、復元機能はステップ3で実装）
4. しかし、Local Storageには保存されたままであることを確認

動作確認チェックリスト

このステップを完了する前に、以下を確認してください：

- ☐ **定数定義**： `STORAGE_KEY` に `"taskList"` が正しく設定されている
- ☐ **関数実装**： `saveTasksToStorage()` 関数が正しく実装されている
- ☐ **自動保存**： タスク追加時に `saveTasksToStorage()` が呼び出されている
- ☐ **コンソール確認**： タスク追加時にコンソールに保存完了メッセージが表示される
- ☐ **localStorage確認**： 開発者ツールのLocal Storageに JSON形式のデータが保存されている

次のステップへ

🎉 **おめでとうございます！** タスクの自動保存機能が実装できました。

現在の状況:

- タスクを追加すると **localStorage に保存される** ✅
- しかし、ブラウザ更新時に **画面には表示されない** ❌

次のステップでは、ページ読み込み時にlocalStorage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しょう！

ステップ3で学ぶこと

- ページ初期化パターンの実装
- JSON.parse()を使ったデータ復元
- DOMContentLoaded イベントの活用

> **関連する補足資料で理解を深める:**
>
> - 補足資料: JSON操作詳解 - JSON変換の詳細とエラー対策
> - 補足資料: try-catch詳解 - エラーハンドリングの実践パターン
> - 補足資料: ページ初期化パターン - 次のステップで使用する初期化処理