# 画像整理アプリ ファイル構成案  
  
プロジェクトのルートディレクトリ直下に以下のフォルダ・ファイルを配置することを提案します。

ImageCleanupAssistant/

├── src/ # アプリケーションのソースコード

│ ├── main/ # Electronのメインプロセス関連

│ │ └── main.js # メインプロセスのエントリーポイント (既存のmain.jsをここに移動)

│ │

│ ├── preload/ # プリロードスクリプト関連

│ │ └── preload.js # プリロードスクリプト (既存のpreload.jsをここに移動)

│ │

│ ├── renderer/ # レンダラープロセス関連 (UIとフロントエンドロジック)

│ │ ├── html/

│ │ │ └── index.html # メインのHTMLファイル (既存のindex.htmlをここに移動)

│ │ ├── js/

│ │ │ ├── renderer.js # UIのメインスクリプト (既存のrenderer.jsをここに移動)

│ │ │ ├── confirmationDialog.js # 確認ダイアログのスクリプト (既存のconfirmationDialog.jsをここに移動)

│ │ │ └── components/ # UIコンポーネントごとのJSファイル (例: imageGallery.js, settingsPanel.jsなど)

│ │ │ └── utils/ # ユーティリティ関数 (例: fileHelper.js, domUtils.jsなど)

│ │ ├── css/

│ │ │ └── style.css # スタイルシート (既存のstyle.cssをここに移動)

│ │ │ └── components/ # コンポーネントごとのCSSファイル (任意)

│ │ └── assets/

│ │ ├── icons/ # アプリケーションアイコン、UIアイコンなど

│ │ └── images/ # UI内で使用する画像素材 (例: チュートリアル画像など)

│ │

│ └── common/ # メインプロセスとレンダラープロセスで共有する可能性のあるモジュール

│ └── constants.js # 定数定義など

│

├── docs/ # ドキュメント関連 (既存のdocsフォルダをそのまま利用)

│ ├── UI\_UXデザイン詳細検討タスクリスト.docx

│ ├── 画像整理アプリ アイコンアセット集.docx

│ ├── 画像整理アプリ アイコン選定案.docx

│ ├── 画像整理アプリ デザイン原則案.docx

│ ├── 画像整理アプリ 状態管理・永続化定義.docx

│ ├── 画像整理アプリ 要件定義書.docx

│ └── 画像整理アプリ 詳細インタラクション定義.docx

│

├── mockups/ # モックアップ関連 (既存のmockupsフォルダをそのまま利用)

│ ├── エラータブ.html

│ ├── ブレ画像モックアップ.html

│ ├── レスポンシブ.html

│ ├── 初回起動時ガイダンス.html

│ ├── 削除移動操作 確認ダイアログ.html

│ ├── 設定画面.html

│ └── 類似画像モックアップ.html

│

├── build/ # ビルド後の成果物を格納するフォルダ (自動生成されることが多い)

│

├── tests/ # テストコード

│ ├── unit/ # ユニットテスト

│ └── e2e/ # E2Eテスト

│

├── package.json # プロジェクト定義ファイル (既存のものをそのまま利用)

├── package-lock.json # 依存関係ロックファイル (既存のものをそのまま利用)

└── README.md # プロジェクトの説明ファイル

## 各ディレクトリの説明  
  
\* \*\*`src/`\*\*: アプリケーションの主要なソースコードを格納します。  
 \* \*\*`main/`\*\*: Electronのメインプロセスに関するコードを配置します。主にウィンドウの作成やシステムイベントの処理など、バックグランド側の処理を担当します。  
 \* \*\*`preload/`\*\*: メインプロセスとレンダラープロセス間で安全に情報をやり取りするためのプリロードスクリプトを配置します。  
 \* \*\*`renderer/`\*\*: アプリケーションのUI（HTML、CSS）と、UIを操作するJavaScript（レンダラープロセス）を配置します。  
 \* `html/`: HTMLファイルを格納します。  
 \* `js/`: レンダラープロセスで動作するJavaScriptファイルを格納します。  
 \* `components/`: 再利用可能なUIコンポーネント（例：画像ギャラリー、設定パネルなど）のJavaScriptファイルを機能ごとに分割して格納します。  
 \* `utils/`: 汎用的な補助関数などを格納します。  
 \* `css/`: スタイルシートを格納します。  
 \* `components/`: (任意) コンポーネントごとのCSSファイルを作成する場合に利用します。  
 \* `assets/`: 画像ファイルやアイコンなど、UIで使用する静的リソースを格納します。  
 \* \*\*`common/`\*\*: (任意) メインプロセスとレンダラープロセスの両方から参照される可能性のある共通のコード（定数定義など）を配置します。  
  
\* \*\*`docs/`\*\*: 設計書、要件定義書、デザイン関連ドキュメントなど、プロジェクトに関するドキュメントを格納します。既存のファイルをそのまま活用できます。  
  
\* \*\*`mockups/`\*\*: HTMLモックアップやデザインカンプなどを格納します。既存のファイルをそのまま活用できます。  
  
\* \*\*`build/`\*\*: (将来的に) Electron Builderなどでアプリケーションをパッケージングした際の成果物を格納するディレクトリです。通常はビルドプロセスによって自動生成されます。  
  
\* \*\*`tests/`\*\*: (将来的に) アプリケーションのテストコードを格納します。ユニットテストやE2Eテストなど、種類に応じてサブディレクトリを作成すると管理しやすくなります。  
  
\* \*\*ルートディレクトリ直下のファイル\*\*:  
 \* `package.json`: プロジェクトの情報や依存ライブラリを定義します。  
 \* `package-lock.json`: 依存ライブラリのバージョンを固定します。  
 \* `README.md`: プロジェクトの概要、セットアップ方法、使い方などを記述します。  
  
## 既存ファイルの移動例  
  
\* `main.js` → `src/main/main.js`  
\* `preload.js` → `src/preload/preload.js`  
\* `index.html` → `src/renderer/html/index.html`  
\* `renderer.js` → `src/renderer/js/renderer.js`  
\* `confirmationDialog.js` → `src/renderer/js/confirmationDialog.js` (または `src/renderer/js/components/confirmationDialog.js` など、より詳細な分類も可能です)  
\* `style.css` → `src/renderer/css/style.css`  
  
この構成案はあくまで一例です。プロジェクトの規模やチームの慣習に合わせて調整してください。  
特に `src/renderer/js/` 配下は、機能が増えるにつれてファイル数が多くなるため、`components/` や `views/` (ページ単位のスクリプト)、`services/` (API連携など) といった形でさらに細分化することも有効です。