

Aerune (エアルネ) システム設計概要

1. アプリケーションの基本構成

- ・ **プラットフォーム:** Electron (Windows / Mac用デスクトップアプリ)
- ・ **フロントエンド:** Vanilla JavaScript (ReactやVueなどのフレームワークは不使用) + HTML/CSS
- ・ **通信API:** @atproto/api (Bluesky公式パッケージ)
- ・ **アーキテクチャ方針:** 軽量・高速化を重視。DOMの直接操作と、イベント委譲 (Event Delegation)、フラグメント (DocumentFragment) を駆使してレンダリングの負荷を抑えている設計。

2. 各JavaScriptファイルの役割 (モジュール構成)

通信・データ取得レイヤー

- ・ **bsky-api.js (APIラッパー)**
 - @atproto/api の BskyAgent を使いやすくラップしたクラス。
 - ログイン、ポスト取得、いいね・リポストの送信、通知取得など、Bluesky サーバーとの通信はすべてここを通る。
- ・ **view-loader.js (データ構築・表示指示)**
 - APIから取得した生のデータを「画面にどう出すか」整理するクラス。
 - タイムラインのスレッド化 (親ポストを辿って会話ツリーを構築する処理) や、通知一覧の生成、プロフィール画面の読み込みを担当。

描画・UI構築レイヤー

- ・ **post-renderer.js (ポスト生成・レンダリング)**
 - 1つのポストをHTML (DOM要素) として組み立てる職人。
 - 画像の埋め込み、引用ポストの展開、NSFWのぼかし判定、相対時間 (〇分前) の計算とキャッシュ表示を行う。

- **utils.js (便利ツール群)**
 - リンクの自動生成 (linkify) 、リッチテキストの変換 (renderRichText) 、画像のローカル圧縮 (compressImage) など、汎用的な処理をまとめた裏方ファイル。

ユーザー操作・アクションレイヤー

- **actions.js (ポストに対するアクション)**
 - 「いいね」 「リポスト」 「ブックマーク」 「フォロー」 「ブロック」 など、ボタンを押したあとの裏側のAPI処理を担当するクラス。
- **navigation.js (画面遷移・履歴管理)**
 - 「戻る」 ボタンのために、ユーザーが辿った画面の履歴（スタック）と、その時のスクロール位置を記憶しておくルーター。

コントローラー

- **renderer.js (メインエントリーポイント)**
 - アプリ起動時に一番最初に走り、他の全モジュールを束ねるリーダー。
 - グローバル状態（ログイン中のアカウント、入力中のテキスト、選択された画像など）の管理。
 - クリックイベントの一括監視（イベント委譲）、画面の切り替え（switchView）、投稿の送信処理（sendPost）を行う。

静的データ

- **constants.js (定数・翻訳・アイコン)**
 - 多言語化（i18n）のテキストデータや、SVGアイコンのHTML文字列をキャッシュしておく倉庫。コードの肥大化を防ぐ役割。

3. Aeruneのコアとなる仕組み（重要な設計）

① イベント委譲（Event Delegation）による軽量化

ポストの「いいねボタン」や「画像」一つ一つにクリックイベント (addEventListener) を付けると、タイムラインをスクロールするだけでメモリを爆食してしまう。

そのため、Aeruneでは **画面全体 (document)** でクリックを1箇所だけで監視 し、クリックされた要素に `data-act="like"` のような属性がついたら処理を発火させる、という非常に軽い設計にしている (renderer.js の `installDelegates`)

② 楽観的UI更新 (Optimistic UI)

「いいね」や「リポスト」、ゴミ箱ボタンを押した時、サーバーからの「成功しました」という返事を待たずに、ローカル側で一瞬でボタンの色を変えたり、ポストを非表示にしたりする仕組み。

これにより、APIのレスポンス速度に依存せず、ブラウザ版以上のサクサク感を出している (renderer.js の `switch (act)` 内で実装)

③ スレッド化タイムライン (Threaded Timeline)

タイムラインに流れてくるバラバラのリプライを、APIの取得限界まで上に辿り、「親 → 子 → 孫」の順番に配列を組み直してから画面に出力するアルゴリズム。

さらに「フォロー外の無関係な会話は弾く」「自分宛ては残す」といった高度なフィルターを view-loader.js の `fetchTimeline` 内で行っている。