シュワっとstudy：技術リファクタリング解説書

1. はじめに

本資料は、手話学習支援アプリ「シュワっとstudy」の開発における、AIを用いたコード再構築（リファクタリング）の詳細をまとめたものです。単に機能を実装するだけでなく、**「保守性」「可読性」「ユーザー体験（UX）」**をいかに高めたかに焦点を当てています。

設計の基本指針

- **命名規則**: Airbnbスタイルガイドを参考に、ESLintでチェック可能な**camelCase**（例: playVideo, updateUI）を採用。
- **定数、変数の分離**:
- **state**: データの状態を一元管理。
- **el**: DOM要素（見た目）を整理。

2. 第1回リファクタリング：設計の構造化

最初のフェーズでは、バラバラだったデータを「状態管理（State）」という概念で集約しました。

① Stateによる一元管理

以前は「現在の問題番号」や「ファイルリスト」が独立して存在していましたが、これらを一つのオブジェクトにまとめました。

- **改善前**: 変数が分散しており、状態の変化を追跡するのが困難。
- **改善後**: console.log(state) を確認するだけで、アプリの現在の状況を一目で把握可能。

② メモリ管理の最適化

URL.createObjectURL で生成した一時的なデータは、使用後に URL.revokeObjectURL で明示的に消去するように実装。

- **目的**: スマートフォンのメモリ不足によるクラッシュを防ぐ、実用レベルの必須スキルを導入。

3. 第2回リファクタリング：機能拡張とロジックの洗練

第2段階では、デザインの刷新に合わせて「戻るボタン」の実装と、UI更新の自動化を行いました。

①「戻るボタン」の数学的ロジック

「1つの問題を2回ずつ連続再生する」という特殊な仕様下で、的確に一つ前の問題へ戻るための計算式を導入しました。

- **計算ロジック**:

```
const step = (currentIndex % 2 === 0) ? 2 : 3; [span_8](start_span)// 偶数回目か奇数回目かでステップ数を分岐[span_8](end_span)
state.currentIndex = (currentIndex - step + length) % length; [span_9](start_span)// 最初の問題で戻っても最後へワープ可能[span_9](end_span)
```

② UI更新の自動化（updateUI関数）

「ボタンの表示/非表示」や「テキストの変更」を各所に分散させず、updateUI() という心臓部となる関数に集約しました。

- **メリット**: 状態（一時停止中か、再生中か等）に応じて画面が自動的に同期されるため、表示の矛盾（バグ）を根絶。

4. 採用した技術・アルゴリズムの詳細

① flatMap: リスト作成の効率化

2回連続再生を実現するために、配列を加工しながら展開する flatMap を採用しました。

- **役割**: [問1, 問2] を一度 [[問1, 問1], [問2, 問2]] に拡張し、即座に [問1, 問1, 問2, 問2] というフラットな1本のリストへ変換します。
- **理由**: 通常の map では二次元配列（箱の中に箱がある状態）になり、再生順の制御が複雑化するためです。

② 三項演算子（疑問符演算子）

if...else 文を簡潔に記述するショートカットを採用。

- **活用例**: isPaused ? [span_16](start_span)[span_17](start_span)'flex' : 'none' のように記述することで、コードの意図を明確に保ちながら行数を削減しています。

③ Fisher-Yates シャッフル

動画の並べ替えには、数学的に公平な「Fisher-Yates（フィッシャー・イェーツ）」アルゴリズムを使用しました。

- **重要性**: Math.random() の単純なソートで発生する「混ざり具合の偏り」を排除し、信頼性の高いランダム性を保証します。

5. デザイン・フォント戦略

視認性を高めるため、英字フォントに 『**Philosopher**』 を採用しました。

見やすいフォント選びの3ルール

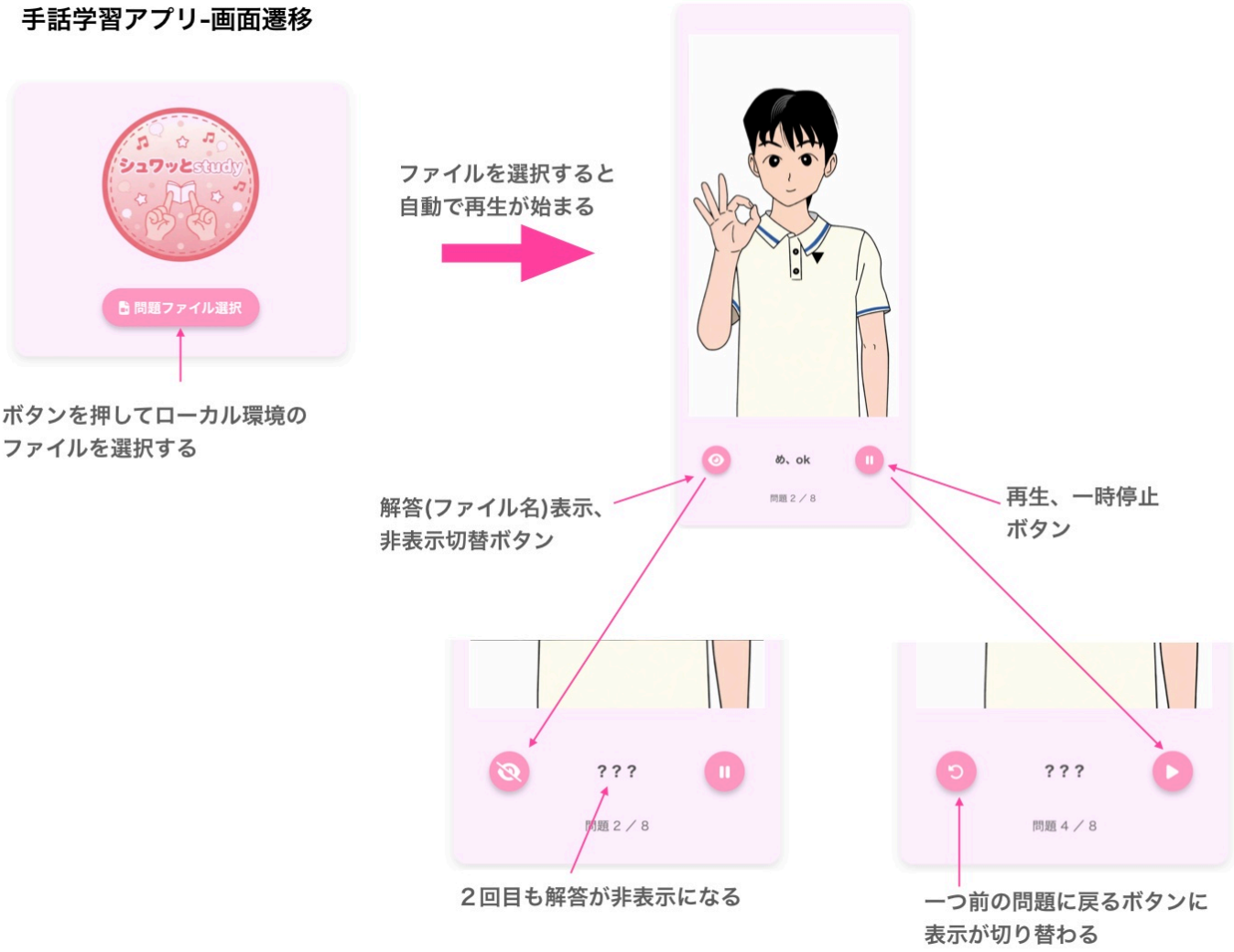
① **高x-height**: 小文字の高さ（x-height）が高いものを選び、スマホの小画面での識別性を向上。

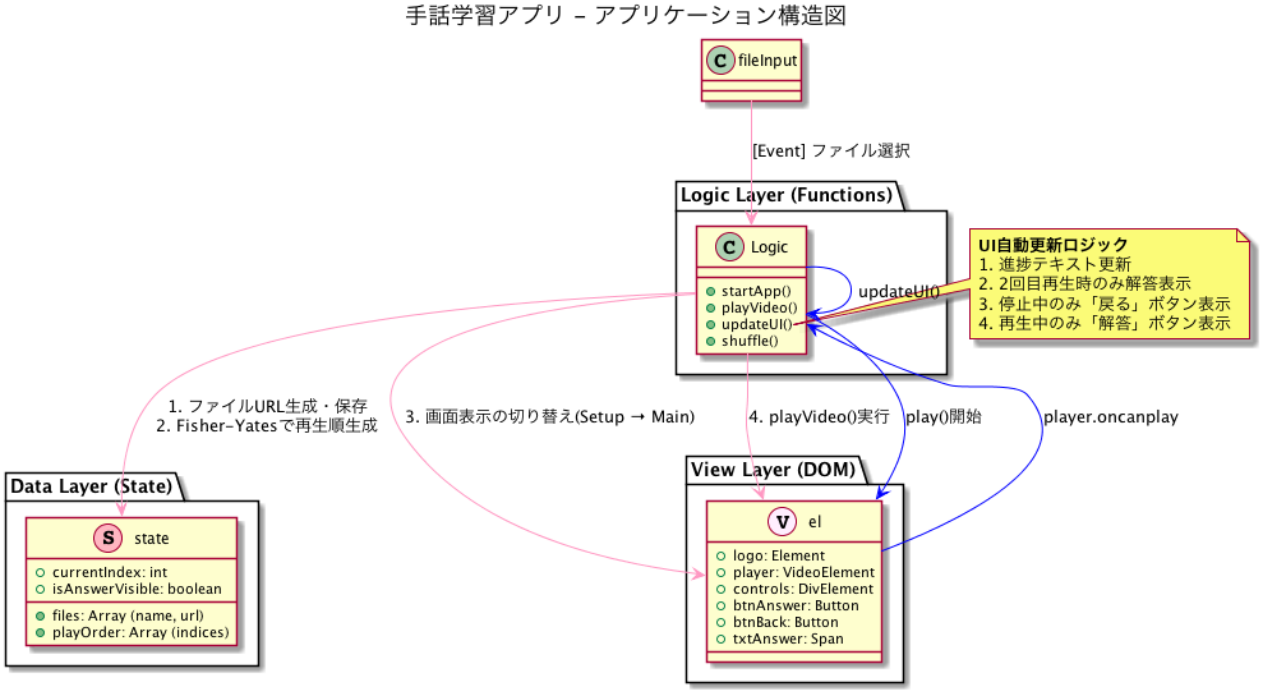
② セリフ・サンセリフの使い分け:

- **タイトル**: 印象に残る「セリフ体」で独自性を強調。
- **本文**: 読みやすさを重視した「サンセリフ体」。

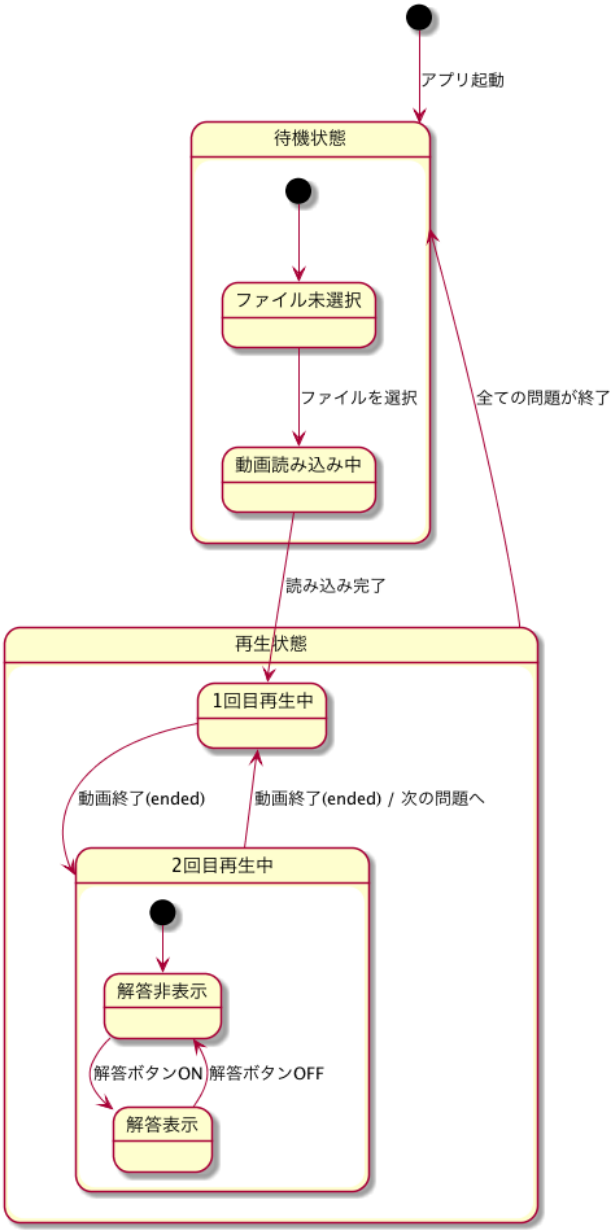
③ ウェイトのコントラスト: 細字と太字を明確に使い分け、情報の優先順位を直感的に伝達。

4. UML等参考図


手話学習アプリ-画面遷移
シュワッとstudy
問題ファイル選択
ボタンを押してローカル環境の
ファイルを選択する
ファイルを選択すると
自動で再生が始まる
め、ok
問題２／８
解答(ファイル名)表示、
非表示切替ボタン
再生、一時停止
ボタン
？？？
問題２／８
２回目も解答が非表示になる
？？？
問題４／８
一つ前の問題に戻るボタンに
表示が切り替わる


手話学習アプリ – アプリケーション構造図
fileInput
[Event] ファイル選択
Logic Layer (Functions)
Logic
startApp()
playVideo()
updateUI()
shuffle()
updateUI()
UI自動更新ロジック
1. 進捗テキスト更新
2. 2回目再生時のみ解答表示
3. 停止中のみ「戻る」ボタン表示
4. 再生中のみ「解答」ボタン表示
1. ファイルURL生成・保存
2. Fisher–Yatesで再生順生成
3. 画面表示の切り替え(Setup → Main)
4. playVideo()実行
play()開始
player.oncanplay
Data Layer (State)
state
currentIndex: int
isAnswerVisible: boolean
files: Array (name, url)
playOrder: Array (indices)
View Layer (DOM)
el
logo: Element
player: VideoElement
controls: DivElement
btnAnswer: Button
btnBack: Button
txtAnswer: Span

手話学習アプリ － 画面遷移図



手話学習アプリ － 再生フロー