View画面作成手順ガイド

本ドキュメントは、Vue 3 + Composition API を使ったSPA開発において、
画面コンポーネントの作成手順を体系的にまとめたものです。

1. 作成の基本手順

1.1 ファイル作成

- `/src/views/` 配下に画面ごとの `.vue` ファイルを作成する
- ファイル名はパスカルケース推奨（例：`UserList.vue`, `UserEdit.vue`）

1.2 テンプレート基本構成

```
<template>
  <div class="p-4">
    <!-- 画面レイアウト・フォーム・一覧などをここに -->
  </div>
</template>

<script setup lang="ts">
import { ref, reactive, onMounted } from 'vue';
</script>

<style >
/* 必要に応じてスタイルを追加 */
</style>
```

2. Composition API基本パターン

Composition APIでは、状態管理・ライフサイクル処理・イベントハンドラを
すべて `script setup` 内でシンプルに管理します。

2.1 状態管理（State）

- フォームの単一フィールド → `ref`
- フォームの複数フィールド（オブジェクト）→ `reactive`

例：

```
const userName = ref(''); // 単一フィールド
const form = reactive({
  email: '',
  password: ''
}); // 複数フィールド
```

2.2 ライフサイクル処理

コンポーネントの生成・描画・更新・破棄に応じて、
適切なライフサイクルイベントを利用します。

主要なライフサイクルイベント

イベント	タイミング	典型的な用途	使用例
`onBeforeMount`	DOMに描画される直前	初期化（最低限の設定）	ローディングフラグON
`onMounted`	DOMに描画された直後	APIデータ取得、初回レンダリング処理	初期データロード
`onBeforeUpdate`	DOMが再描画される直前	更新前に状態を保存	編集内容の一時保存
`onUpdated`	DOMが再描画された直後	再描画後の後処理	スクロール位置リセット
`onBeforeUnmount`	コンポーネント破棄直前	リスナー・タイマー解除	setInterval解除など
`onUnmounted`	コンポーネント破棄後	完全なクリーンアップ処理	ストアリセットなど

使い方パターン例

画面ロード時に初期データ取得（onMounted）

```
onMounted(loadUserList);

const loadUserList = async () => {
  users.value = await getUserList();
};
```

編集開始前に状態を記録（onBeforeUpdate）

```
onBeforeUpdate(() => {
  previousFormData.value = { ...formData.value };
});
```

2.3 API呼び出し