PointMarker 機能仕様書

1. プロジェクト概要

1.1 プロジェクト名

PointMarker（旧RouteMarker/PickPoints）

1.2 目的

ハイキングマップ画像上にポイントとルートをマーキングし、構造化されたJSONデータとして出力するWebアプリケーション

1.3 対象ユーザー

- ハイキング・登山愛好者
- 地理情報管理者
- マップデータ作成者

1.4 技術仕様

- **言語**: バニラJavaScript（ES6モジュール）
- **UI**: HTML5、CSS3、Canvas API
- **ファイル処理**: File System Access API
- **データ形式**: JSON
- **レスポンシブ対応**: CSS Flexbox、CSS Grid

2. アーキテクチャ

2.1 フォルダ構造

```
PointMarker/
├── index.html                  # メインHTMLファイル
├── styles.css                  # スタイルシート
├── CLAUDE.md                   # プロジェクト指針
├── README.md                   # プロジェクト概要
├── docs/                       # ドキュメント
│   ├── funcspec.md             # 機能仕様書
│   └── UsersGuide-202508.md    # ユーザーガイド
└── js/                         # JavaScriptモジュール
    ├── app.js                  # メインアプリケーション
    ├── core/
    │   └── Canvas.js           # キャンバス描画管理
    ├── data/
    │   ├── FileHandler.js      # ファイル操作
    │   ├── PointManager.js     # ポイント管理
    │   └── RouteManager.js     # ルート管理
    ├── ui/
    │   ├── InputManager.js     # 動的入力管理
```

```
    │   └── LayoutManager.js     # レイアウト管理
    └── utils/
        ├── Coordinates.js       # 座標変換
        └── Validators.js        # バリデーション
```

2.2 設計パターン

- **モジュール分離**: ES6モジュールによる機能別分離
- **MVC風アーキテクチャ**: データ・UI・ビジネスロジック分離
- **コールバック駆動**: 疎結合なコンポーネント間通信
- **レスポンシブデザイン**: CSS変数とFlexboxによる柔軟なレイアウト

2.3 主要クラス構成

メインクラス

- **PointMarkerApp**: アプリケーション統合管理
- **CanvasRenderer**: キャンバス描画管理

データ管理層

- **PointManager**: ポイントデータ管理と永続化
- **RouteManager**: ルートデータ管理と検証
- **FileHandler**: ファイル操作とFile System Access API統合

UI管理層

- **InputManager**: 動的入力フィールド管理
- **LayoutManager**: レイアウト・モード状態管理

ユーティリティ層

- **CoordinateUtils**: 座標系変換処理
- **Validators**: データ検証とフォーマット処理

3. 機能仕様

3.1 画像管理機能

3.1.1 画像読み込み

- **対応形式**: PNG形式
- **読み込み方法**: File System Access API（フォールバック：従来のinput要素）
- **表示**: HTML5 Canvasによるレスポンシブ表示
- **座標系管理**: 画像座標とキャンバス座標の自動変換

3.1.2 画像表示制御

- **自動スケーリング**: 表示領域に合わせた自動リサイズ

- **アスペクト比維持**: 元画像の比率保持
- **レスポンシブ対応**: ウィンドウサイズ変更時の自動調整

実装メソッド

- `PointMarkerApp.handleImageSelection()`: File System Access API画像選択
- `PointMarkerApp.processLoadedImage()`: 画像読み込み完了処理
- `CanvasRenderer.setImage()`: 描画対象画像設定
- `CanvasRenderer.setupCanvas()`: キャンバスサイズ調整

3.2 レイアウト管理機能

3.2.1 レイアウトモード

- **サイドバーモード**: 横並びレイアウト（デフォルト）
- **オーバーレイモード**: 地図上オーバーレイ表示
- **動的切り替え**: リアルタイムレイアウト変更

3.2.2 編集モード

- **ポイント編集モード**: ポイント追加・編集・削除
- **ルート編集モード**: ルート中間点追加・開始終了ポイント設定

実装メソッド

- `LayoutManager.setLayout()`: レイアウト変更
- `LayoutManager.setEditingMode()`: 編集モード変更
- `LayoutManager.updateLayoutDisplay()`: レイアウト表示更新

3.3 ポイント編集機能

3.3.1 ポイント操作

- **追加**: キャンバスクリック→動的入力ボックス生成
- **削除**: Escapeキー、または空入力でblur
- **移動**: ドラッグ＆ドロップ（ルート編集モード時は無効）
- **ID編集**: インライン編集、リアルタイム検証

3.3.2 ポイントID管理

- **フォーマット**: X-nn形式（英大文字1桁-数字2桁）
- **自動補正**: 全角→半角変換、0埋め処理
- **入力制御**: 入力中は補正なし、blur時に補正実行
- **バリデーション**: リアルタイム形式検証、エラーフィードバック

3.3.3 一括操作

- **全ポイントクリア**: 確認なし即座削除
- **ポイントID名補正**: 全ポイント一括フォーマット＋空ポイント削除

- **JSON出力**: ポイントデータのJSON形式保存
- **JSON読み込み**: 既存JSONファイルからポイント復元

実装メソッド

- `PointManager.addPoint()`: ポイント追加
- `PointManager.updatePointId()`: ポイントID更新
- `PointManager.formatAllPointIds()`: 全ポイント一括補正
- `InputManager.createInputBox()`: 動的入力ボックス生成
- `InputManager.positionInputBox()`: 最適位置計算

3.4 ルート編集機能

3.4.1 ルート構成要素

- **開始ポイント**: 既存ポイントIDから選択（自動補正あり）
- **終了ポイント**: 既存ポイントIDから選択（自動補正あり）
- **中間点**: キャンバスクリック→ルートポイント追加

3.4.2 ルート編集制限

- **ポイント編集禁止**: 既存ポイントの移動・削除を制限
- **入力フィールド無効化**: 背景色変更（#e0e0e0）
- **開始終了ポイント強調**: 指定ポイントの背景白色化＋青枠表示

3.4.3 ルート操作

- **中間点追加**: 順次クリック→ルートポイント蓄積
- **中間点クリア**: 全中間点一括削除
- **ルートJSON出力**: ルート専用JSON形式で保存
- **ルートJSON読み込み**: 既存ルートJSONから復元

実装メソッド

- `RouteManager.addRoutePoint()`: 中間点追加
- `RouteManager.setStartPoint()` / `setEndPoint()`: 開始終了ポイント設定
- `RouteManager.validateStartEndPoints()`: ポイント存在検証
- `InputManager.setHighlightedPoints()`: ポイント強調表示

3.5 データ検証機能

3.5.1 ポイント検証

- **ID形式検証**: X-nn形式の厳密チェック
- **重複ID検証**: 同一ID存在チェック
- **空ポイント検出**: ID未入力ポイントの自動検出・削除

3.5.2 ルート検証

- **開始ポイント存在確認**: 指定IDがポイントとして存在するか検証
- **終了ポイント存在確認**: 指定IDがポイントとして存在するか検証
- **中間点数確認**: 最低1つ以上の中間点存在チェック
- **必須項目確認**: 開始・終了ポイント両方の設定確認

実装メソッド

- `Validators.isValidPointIdFormat()`: ポイントID形式検証
- `Validators.formatPointId()`: ポイントID自動補正
- `RouteManager.validateStartEndPoints()`: 総合ルート検証

3.6 ファイル操作機能

3.6.1 画像ファイル処理

- **PNG読み込み**: File System Access API優先、フォールバック対応
- **ファイル形式検証**: MIME typeによるPNG形式確認
- **エラーハンドリング**: 不正ファイル読み込み時のエラー表示

3.6.2 JSON処理

- **ポイントJSON**: ポイント専用データ構造での入出力
- **ルートJSON**: ルート専用データ構造での入出力
- **ファイル名自動生成**:
  - ポイント: `{画像名}_points.json`
  - ルート: `{画像名}_route_{開始ポイント}_to_{終了ポイント}.json`

実装メソッド

- `FileHandler.selectImage()`: File System Access API画像選択
- `FileHandler.saveJSONWithUserChoice()`: JSON保存
- `PointManager.exportToJSON()`: ポイントJSON生成
- `RouteManager.exportToJSON()`: ルートJSON生成

4. データ構造

4.1 ポイントJSON形式

```
{
    "totalPoints": 3,
    "imageReference": "sample.png",
    "imageInfo": {
        "width": 1920,
        "height": 1080
    },
    "points": [
        {
            "index": 1,
            "id": "A-01",
```

```
            "imageX": 245,
            "imageY": 387,
            "isMarker": false
        }
    ],
    "exportedAt": "2025-08-24T10:30:00.000Z"
}
```

4.2 ルートJSON形式

```
{
    "routeInfo": {
        "startPoint": "A-01",
        "endPoint": "B-03",
        "waypointCount": 5
    },
    "imageReference": "sample.png",
    "imageInfo": {
        "width": 1920,
        "height": 1080
    },
    "points": [
        {
            "type": "waypoint",
            "index": 1,
            "imageX": 320,
            "imageY": 450
        }
    ],
    "exportedAt": "2025-08-24T10:45:00.000Z"
}
```

4.3 座標系管理

- **画像座標系**: 元PNG画像の実際のピクセル座標（永続化用）
- **キャンバス座標系**: 表示用にスケールされた座標（描画用）
- **スクリーン座標系**: ブラウザ内の絶対位置座標（UI配置用）
- **マウス座標系**: ブラウザイベントから得られる座標

5. UI/UX仕様

5.1 レスポンシブデザイン

- **ブレークポイント**: 768px（モバイル対応境界）
- **レイアウト**: Flexboxベースの柔軟な配置
- **フォント**: システムフォント優先、日本語対応

5.2 アクセシビリティ

- **キーボード操作**: Tab移動、Escape削除対応
- **ARIA属性**: スクリーンリーダー対応
- **カラーコントラスト**: WCAG準拠の配色
- **フォーカス管理**: 視覚的フォーカスインジケーター

5.3 ビジュアル仕様

- **カラーパレット**: CSS変数による統一配色
- **ボタンデザイン**: 機能別カラーコーディング
- **フィードバック**: ホバー効果、状態変化アニメーション
- **エラー表示**: 赤色背景による入力エラー表示

5.4 動的UI要素

- **入力ボックス**: ポイント位置に動的配置
- **位置最適化**: 画面端を考慮した自動位置調整
- **状態管理**: モード切り替えに応じた表示制御
- **リアルタイム更新**: 入力値変更の即座反映

6. パフォーマンス仕様

6.1 描画パフォーマンス

- **Canvas最適化**: 必要時のみ再描画
- **座標キャッシュ**: スケール計算結果の再利用
- **イベント効率化**: デバウンス処理によるリサイズ最適化

6.2 メモリ管理

- **オブジェクト再利用**: 不要なオブジェクト生成回避
- **イベントリスナー管理**: 適切な削除とメモリリーク防止
- **画像メモリ**: 大容量画像対応

7. ブラウザ対応

7.1 対応ブラウザ

- **Chrome**: 86+ (File System Access API対応)
- **Firefox**: 最新版（フォールバック動作）
- **Safari**: 14+（フォールバック動作）
- **Edge**: 86+（Chromiumベース）

7.2 必要な機能

- **ES6モジュール**: import/export構文
- **Canvas API**: 2D描画機能
- **File API**: ファイル読み込み
- **File System Access API**: 高度なファイル操作（オプション）

8. セキュリティ仕様

8.1 ファイルアクセス

- **同一オリジン制限**: ローカルファイルアクセス制限
- **ファイル形式検証**: MIME typeによる安全性確認
- **サニタイゼーション**: 入力値の適切な処理

8.2 データ保護

- **ローカル処理**: サーバー送信なし、プライバシー保護
- **XSS対策**: 動的コンテンツの適切なエスケープ
- **CSP対応**: Content Security Policy準拠

9. 拡張性

9.1 モジュール拡張

- **プラグイン機構**: コールバックベースの拡張ポイント
- **カスタムバリデーター**: Validators拡張
- **出力形式拡張**: JSON以外のフォーマット対応可能性

9.2 機能拡張候補

- **GPS連携**: 位置情報との統合
- **マルチレイヤー**: 複数画像の重ね合わせ
- **テンプレート機能**: ポイント配置パターンの保存・再利用

10. テスト仕様

10.1 テスト対象

- **機能テスト**: 各操作の正常動作確認
- **データ整合性**: JSON入出力の正確性検証
- **UI応答性**: レスポンシブ動作の確認
- **ブラウザ互換性**: 対象ブラウザでの動作検証

10.2 テスト方法

- **手動テスト**: ブラウザでの実操作テスト
- **自動テスト**: 将来的なユニットテスト導入可能性
- **回帰テスト**: 機能変更時の既存機能影響確認

最終更新: 2025年8月24日
バージョン: 2.0
作成者: Claude Code Analysis