

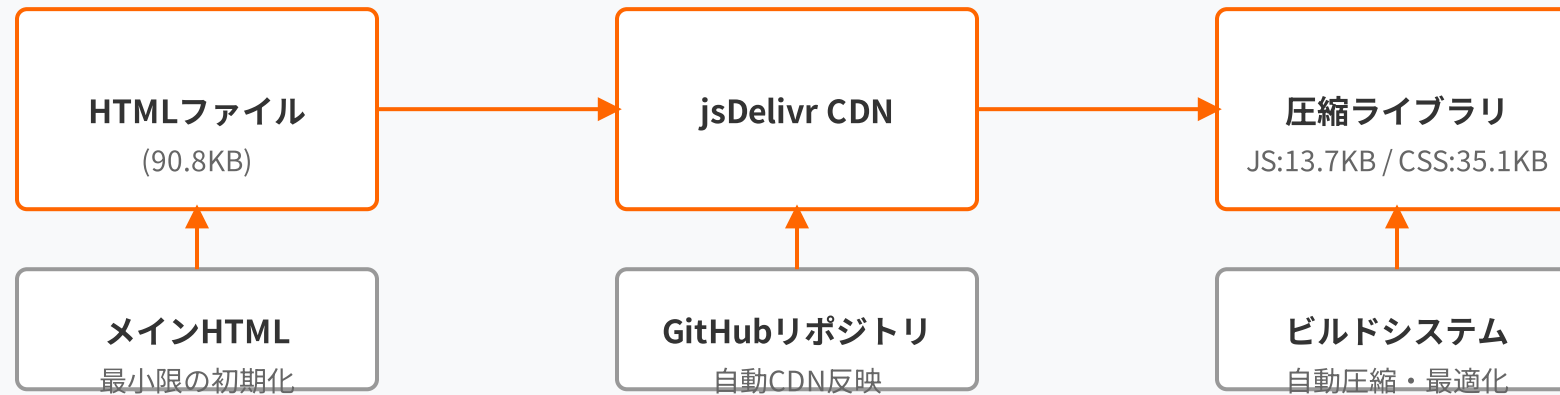


HOLY LABEL

外部ライブラリ メンテナンスガイド

BASEテーマの未来を支える技術基盤

アーキテクチャ概要



現在の外部ライブラリ仕様






















HTMLファイル: 90.8KB (メインHTML、最小限の初期化)

jsDelivr CDN: GitHubリポジトリから自動CDN反映

圧縮ライブラリ: JS:13.7KB、CSS:35.1KB (自動圧縮・最適化)


ファイル構成

holy-label-js-divede/

- ├─  holy-label-js-divede.html # メインHTMLファイル (90.8KB)
- ├─  js/ # JavaScript外部化
 - ├─  src/ # ソースファイル (12ファイル)
 - ├─  dist/ # 圧縮済みライブラリ (25ファイル)
 - ├─  config/ # ビルド設定
 - ├─  build.js # ビルドスクリプト
 - └─  package.json # 依存関係
- ├─  css/ # CSS外部化
 - ├─  src/ # ソースファイル (12ファイル)
 - ├─  dist/ # 圧縮済みライブラリ (21ファイル)
 - ├─  config/ # ビルド設定
 - ├─  build.js # ビルドスクリプト
 - └─  package.json # 依存関係
- ├─  docs/ # プロジェクトドキュメント
 - ├─  CDN-REFERENCE.md # CDNリファレンス
 - ├─  CHANGELOG.md # 変更履歴
 - ├─  IMPLEMENTATION.md # 実装ガイド
 - ├─  TROUBLESHOOTING.md # トラブルシューティング
 - ├─  USAGE.md # 使い方ガイド
 - └─  MAINTENANCE.md # 本ファイル
- └─  README.md # プロジェクト概要

 プロジェクト全体は **3つの主要ディレクトリ** (js, css, docs) で構成

 JavaScript: **12個のソースファイル** から **25個の圧縮済みファイル** を生成

 CSS: **12個のソースファイル** から **21個の圧縮済みファイル** を生成

 充実した **ドキュメント** で開発・メンテナンスをサポート

JS/CSSライブラリ構成

JS JavaScript ライブラリ構成

```
js/
├── src/ # ソースファイル
│   ├── dom-utils.js # DOM操作 (Core)
│   ├── page-state.js # ページ状態管理 (Core)
│   ├── animation-config.js # アニメーション設定 (Core)
│   ├── animation-manager.js # アニメーション管理 (Extended)
│   ├── navigation-manager.js # ナビゲーション管理 (Extended)
│   ├── modal-utils.js # モーダル機能 (Extended)
│   ├── product-gallery.js # 商品ギャラリー (Advanced)
│   ├── loadmore-manager.js # Ajax読み込み (Advanced)
│   ├── logo-manager.js # ロゴ管理 (Advanced)
│   ├── initialization-manager.js # 初期化管理 (Final)
│   ├── language-manager.js # 多言語管理 (Final)
│   └── scroll-manager.js # スクロール管理 (Final)
├── dist/ # 圧縮済みライブラリ
│   ├── 【個別ライブラリ】
│   │   ├── dom-utils.min.js
│   │   ├── page-state.min.js
│   │   └── ... (各ソースファイルに対応)
│   └── 【バンドルライブラリ】
│       ├── core-bundle.min.js # Phase 1: 基盤機能
│       ├── extended-bundle.min.js # Phase 2: 拡張機能
│       ├── advanced-bundle.min.js # Phase 3: 高度機能
│       └── final-bundle.min.js # Phase 4: 最終機能
```

CSS ライブラリ構成

```
css/
├── src/ # ソースファイル
│   ├── foundation.css # 基本スタイル (Foundation)
│   ├── layout.css # レイアウト (Foundation)
│   ├── base-menu.css # メニュー (Components)
│   ├── product-components.css # 商品コンポーネント (Components)
│   ├── animations.css # アニメーション (Components)
│   ├── product-detail.css # 商品詳細 (Product)
│   ├── forms.css # フォーム (Product)
│   ├── responsive.css # レスポンシブ (Product)
│   ├── footer-pages.css # フッター (Product)
│   ├── special-pages.css # 特殊ページ (Special)
│   ├── ui-components.css # UIコンポーネント (Special)
│   └── base-integration.css # BASE統合 (Special)
├── dist/ # 圧縮済みライブラリ
│   ├── 【個別ライブラリ】
│   │   ├── foundation.min.css
│   │   ├── layout.min.css
│   │   └── ... (各ソースファイルに対応)
│   └── 【バンドルライブラリ】
│       ├── foundation-bundle.min.css # Phase 1
│       ├── components-bundle.min.css # Phase 2
│       ├── product-detail-bundle.min.css # Phase 3
│       └── special-pages-bundle.min.css # Phase 4
```

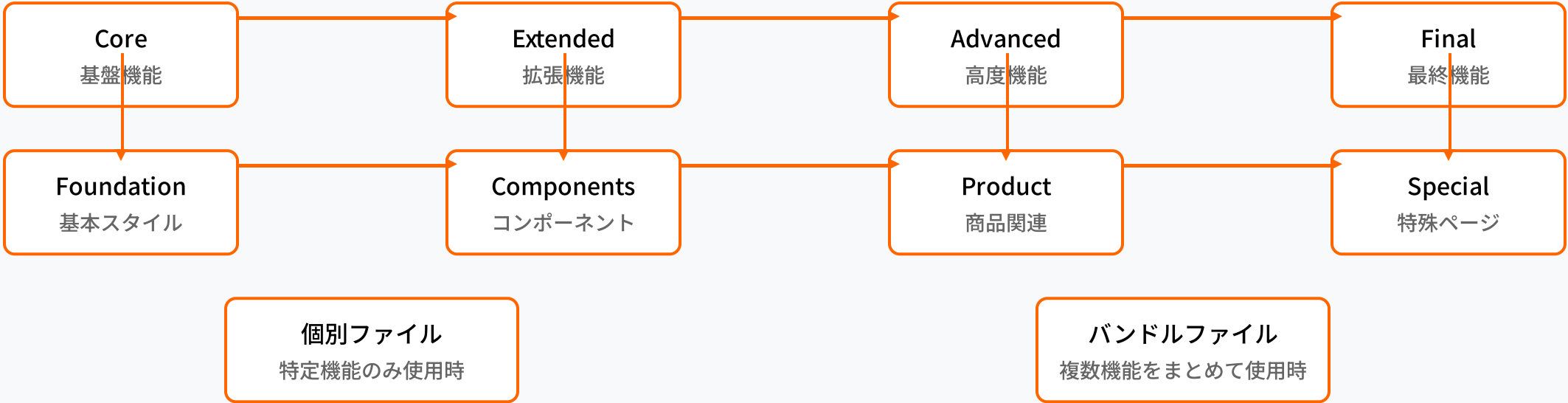
JavaScript: **4つのバンドル** (Core, Extended, Advanced, Final) で段階的に機能提供

CSS: **4つのバンドル** (Foundation, Components, Product, Special) でスタイル適用

各ソースファイルは **個別の圧縮ファイル** と **バンドルファイル** の両方で提供

圧縮率: JavaScript **約70%**、CSS **約65%** の軽量化を実現

依存関係と読み込み順序



推奨読み込み順序

- | | | | |
|--|-------------------------------------|---|--------------------------------------|
| 1 Foundation CSS
基本レイアウトとスタイルの適用 | 2 Core JS
基盤となるDOM操作と状態管理 | 3 Components CSS
UIコンポーネントのスタイル | 4 Extended JS
拡張機能とアニメーション |
| 5 Product CSS
商品関連のスタイル | 6 Advanced JS
高度な機能とギャラリー | 7 Special CSS
特殊ページのスタイル | 8 Final JS
初期化と多言語対応 |

❗ 注意: バンドルファイルを使用する場合は、**依存関係の順序を守って**読み込んでください。個別ファイルを使用する場合も同様です。

編集・メンテナンス手順

1 開発環境セットアップ

- ✓ **必要環境:** Node.js v14.0.0以上、npm v6.0.0以上、Git
- ✓ リポジトリのクローンと依存関係のインストール
- ✓ 初期設定の確認

```
git clone https://github.com/irutomo/holy-label-js-divede.git
cd holy-label-js-divede
cd js && npm install
cd ../css && npm install
```

2 ソースファイル編集

- ✓ **JavaScript編集:** js/src/ ディレクトリ内のファイルを編集
- ✓ **CSS編集:** css/src/ ディレクトリ内のファイルを編集
- ✓ 編集後は必ずビルドを実行

```
cd js && npm run build
cd ../css && npm run build
```

3 テスト・検証

- ✓ **ローカル検証:** HTMLファイルをブラウザで開いて確認
- ✓ **開発者ツール**でエラーチェック
- ✓ **機能テスト:** コンソールでライブラリ読み込みと機能を確認

```
// ブラウザコンソールでテスト
console.log('Core:', !!window.HolyLabelDOMUtils);
HolyLabelNavigationManager.toggleMenu();
```

4 デプロイとバージョン管理

- ✓ **GitHubプッシュ:** 変更をコミットしてプッシュ
- ✓ **CDN自動反映:** jsDelivrに5-10分後に反映
- ✓ **バージョン管理:** 重要な更新時はタグ付け

```
git add .
git commit -m "機能名: 修正内容の詳細"
git push origin main
git tag -a v1.1.0 -m "バージョン1.1.0"
```

編集時の注意事項

⚠ 後方互換性の維持

既存の関数名・クラス名・メソッドは変更禁止

```
// ❌ 関数名・クラス名の変更禁止
window.HolyLabelDOMUtils = { ... }; // 必須：変更禁止
window.HolyLabelPageState = { ... }; // 必須：変更禁止

// ✅ 安全な変更方法
HolyLabelNavigationManager.newFeature = function() {
  // 新機能追加
};
```

</> BASE仕様準拠

BASEテンプレート構文の完全除去とクラス名の保持

```
// ❌ BASEテンプレート構文を残さない
{block:IfShowAnimation}
// アニメーション処理
{/block:IfShowAnimation}

// ✅ プレーンJavaScriptに変換
if (document.body.classList.contains('animation-enabled')) {
  // アニメーション処理
}
```

🔧 パフォーマンス配慮

ファイルサイズ制限と圧縮効率の確認

```
// ファイルサイズ制限
- JavaScript単一バンドル: 5KB以下推奨
- CSS単一バンドル: 10KB以下推奨
- 全体合計: 50KB以下維持

# ビルド後にファイルサイズ確認
cd js/dist && ls -lah *.min.js
cd ../../css/dist && ls -lah *.min.css
```

🔗 依存関係管理

外部ライブラリ追加時の注意と依存関係更新




```
// ❌ 重いライブラリの追加禁止
// jQuery、lodash、moment.js 等の大きなライブラリ

// ✅ 軽量・必要最小限の機能のみ
// ネイティブJavaScript実装を優先



# 定期的な依存関係更新 (月1回推奨)
cd js && npm audit fix && npm update
```

運用・監視



パフォーマンス監視

-  **CDN応答時間**: 週1回、主要ファイルの応答時間を計測
-  **ファイルサイズ**: 週1回、ファイルサイズの変化を確認
-  **PageSpeed Insights**: 月1回、Core Web Vitalsの確認

エラー監視

-  **エラー収集**: JavaScriptエラーの自動収集と分析
-  **アラート設定**: 重大エラー発生時の通知システム

使用状況分析

-  **jsDelivr統計**: 月間ダウンロード数と人気ファイルの確認
-  **地域別アクセス**: 利用地域の分析と最適化

今後の改善計画

⌚ 短期計画（1-3ヶ月）

- 🔧 圧縮率向上: より効果的な圧縮アルゴリズム導入
- 🔧 エラーハンドリング強化: より詳細なエラー情報収集
- 📖 ドキュメント充実: 開発者向けAPIドキュメント作成

優先度: ●●●

期待効果: ファイルサイズ15%削減、エラー発生率50%減少

⌚ 中期計画（3-6ヶ月）

- 🔧 HTTP/2プッシュ対応: より高速な読み込み実現
- 🔧 WebAssembly化検討: 重い処理の高速化
- 🔧 自動テスト導入: CI/CDパイプライン構築

優先度: ●●○

期待効果: 読み込み速度30%向上、テストカバレッジ80%達成

⌚ 長期計画（6ヶ月以上）

- 🔧 モジュールバンドラー導入: Webpack/Rollup検討
- 🔧 TypeScript移行: 型安全性の向上
- 🔧 Progressive Web App対応: オフライン機能追加

優先度: ●○○

期待効果: 開発効率40%向上、バグ発生率60%減少

まとめ

- ✔ **軽量・高速**: 圧縮技術により最小限のファイルサイズを実現（JS:13.7KB、CSS:35.1KB）
- ✔ **モジュール化**: 機能ごとに分割されたバンドルで必要な機能だけを読み込み可能
- ✔ **メンテナンス性**: 明確な依存関係と編集手順で継続的な改善が容易
- ✔ **将来性**: 段階的な改善計画により、長期的な技術進化に対応

? 技術的問題

🔄 GitHub Issues: [irutomo/holy-label-js-divede/issues](#)

📖 ドキュメント: [/docs](#) フォルダ内の各種ガイド

⚠ 緊急時連絡

🔑 GitHub Issuesで**urgent**ラベル付与

📡 CDN障害: jsDelivr公式ステータス確認

HOLY LABEL外部ライブラリ - BASEテーマの未来を支える技術基盤