Development Guide - Power Components

Comprehensive guide for developers contributing to the Power Components suite.



Prerequisites

Ensure you have the following installed:

Node.js: v16.0.0 or higher
npm: v8.0.0 or higher
Git: Latest version

• **Obsidian:** v1.4.0 or higher (for testing)

• Code Editor: VS Code recommended with TypeScript support

Development Environment Setup

1. Clone and Setup Repository

```
# Clone the repository
git clone https://github.com/memorymusicllc/power.components.git
cd power.components

# Install root dependencies
npm install

# Install dependencies for all plugins
npm run install-all

# Build all components
npm run build-all
```

2. Link to Obsidian Vault

```
# Link plugins to your development vault
npm run link-vault /path/to/your/obsidian/vault

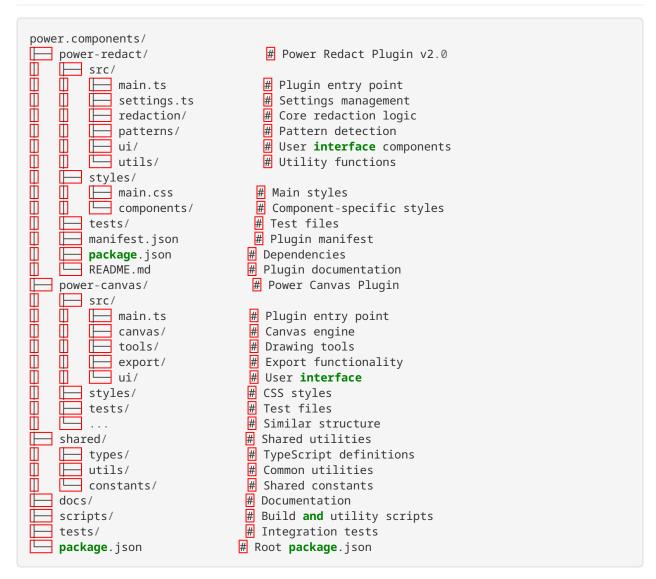
# Or manually create symlinks
In -s $(pwd)/power-redact /path/to/vault/.obsidian/plugins/power-redact
In -s $(pwd)/power-canvas /path/to/vault/.obsidian/plugins/power-canvas
```

3. Start Development Mode

```
# Start development with hot reload
npm run dev

# Or start individual plugins
cd power-redact && npm run dev
cd power-canvas && npm run dev
```

Project Structure



X Development Workflow

Branch Strategy

We use GitFlow branching model:

- main: Production-ready code
- develop: Integration branch for features
- feature/*: Individual feature development
- hotfix/*: Critical bug fixes
- release/*: Release preparation

Creating a Feature Branch

```
# Create feature branch from develop
git checkout develop
git pull origin develop
git checkout -b feature/your-feature-name

# Work on your feature
# ... make changes ...

# Commit changes
git add .
git commit -m "feat: add your feature description"

# Push feature branch
git push origin feature/your-feature-name
```

Code Standards

TypeScript Configuration

```
// tsconfig.json
  "compilerOptions": {
    "target": "ES2020",
    "module": "CommonJS",
    "lib": ["ES2020", "DOM"],
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "forceConsistentCasingInFileNames": true,
    "declaration": true,
    "outDir": "./dist",
    "rootDir": "./src"
  },
  "include": ["src/**/*"],
  "exclude": ["node_modules", "dist", "tests"]
}
```

ESLint Configuration

```
// .eslintrc.json
{
  "extends": [
    "@typescript-eslint/recommended",
    "plugin:@typescript-eslint/recommended-requiring-type-checking"
 ],
  "parser": "@typescript-eslint/parser",
  "parserOptions": {
    "project": "./tsconfig.json"
  },
  "rules": {
    "@typescript-eslint/no-unused-vars": "error",
    "@typescript-eslint/explicit-function-return-type": "warn",
    "prefer-const": "error",
    "no-var": "error"
  }
}
```

Code Style Guidelines

```
1. Naming Conventions:
   ```typescript
 // Classes: PascalCase
 class RedactionEngine {}
// Functions/Variables: camelCase
const processText = () => \{\};
let isEnabled = true;
// Constants: UPPER_SNAKE_CASE
const MAX_PATTERN_LENGTH = 1000;
// Interfaces: PascalCase with 'I' prefix (optional)
interface IPatternDetector {}
 1. File Organization:
   ```typescript
   // Import order: external, internal, relative
   import { Plugin } from 'obsidian';
   import { RedactionEngine } from '../redaction/engine';
   import './styles.css';
// Export order: types, constants, functions, classes
export type RedactionStyle = 'blackout' | 'blur';
export const DEFAULT_PATTERNS = {};
export function sanitizeText() {}
export class PatternDetector {}
 1. Documentation:
    ```typescript
 /**

 Processes text for redaction patterns

 • @param text - The input text to process
 • @param patterns - Array of pattern names to apply
 • @returns Promise resolving to redaction result
 @throws {Error} When pattern compilation fails
 */
 async function processText(
 text: string,
 patterns: string[]
): Promise
```