⚡ Canva App Analysis Report

sample-js-poor-contrast.js

Generated: June 1, 2025 at 12:30 PM

Analysis Duration: 26.73s

📊 Overall Assessment

46

Canva-Ready Score: 46/100

🔍 Analysis complete: limited Canva app readiness with a score of 46/100. Found 18 issues including 2 critical issues requiring immediate attention.

2	6	6	4	18
Critical	High	Medium	Low	Total Issues

📈 Category Breakdown

Security (3 issues)	91/100
Code Quality (7 issues)	61/100
UI & UX (8 issues)	0/100

💡 Key Recommendations

- 🚨 URGENT: This code has critical issues that need immediate attention before deployment.
- 🔒 Security: Excellent standards maintained.
- ⚙️ Code Quality: Good foundation with room for minor improvements.
- 🎨 UI UX: 2 critical issues need immediate fixes.
- 📋 Next Steps: Start by addressing the top 3 highest-priority issues:

🔍 Detailed Findings

📋 All Issues Summary

Total Issues Found: 18

Critical Priority Issues (2)

Multiple WCAG contrast failures throughout app

CRITICAL

Light gray text on light backgrounds fails WCAG 2.1 AA standards, making content unreadable for users with visual impairments.

🔍 Issue Found Here:

```
title.style.color = '#cccccc'; title.style.backgroundColor = '#f0f0f0';
```

💡 Suggestion:

Replace custom colors with Canva's functional color tokens like colorTypographyPrimary for proper contrast ratios.

Custom styling violates Canva design system CRITICAL

Direct style manipulation bypasses Canva's App UI Kit components and design tokens, breaking visual consistency.

🔍 **Issue Found Here:**

```
container.style.backgroundColor = '#f5f5f5';
```

💡 **Suggestion:**

Use Canva's Box, Text, Title, and Button components with functional colors instead of custom CSS.

High Priority Issues (6)

Monolithic function violates single responsibility HIGH

createPoorContrastApp() handles UI creation, styling, event handling, and mounting - should be split into smaller, focused functions

🔍 **Issue Found Here:**

```
function createPoorContrastApp() {
```

💡 **Suggestion:**

Split into separate functions: createTitle(), createButtons(), createSettings(), mountApp()

Extensive code duplication in styling

HIGH

Repeated inline styling patterns throughout the code - no reusable style utilities or constants

🔍 **Issue Found Here:**

```
style.backgroundColor = '#...'; style.color = '#...';
```

💡 **Suggestion:**

Create style constants object and reusable styling functions

Yellow text on white background nearly invisible

HIGH

Description text uses light yellow on white, creating severe readability issues visible in the screenshot.

🔍 **Issue Found Here:**

```
description.style.color = '#ffff99'; description.style.backgroundColor = 'whit
e';
```

💡 **Suggestion:**

Use Canva's Text component with colorTypographyPrimary or colorTypographySecondary.

Poor button contrast creates usability barriers HIGH

Light blue and yellow buttons with light text are barely visible, impacting critical user interactions.

🔍 **Issue Found Here:**

```
addButton.style.backgroundColor = '#b3d9ff'; addButton.style.color = '#f0f0f
0';
```

💡 **Suggestion:**

Replace with Canva's Button component using primary and secondary variants for proper contrast.

Error message has insufficient contrast HIGH

Red text on pink background fails accessibility standards and doesn't use Canva's error messaging patterns.

🔍 **Issue Found Here:**

```
errorMessage.style.backgroundColor = '#ffccdd'; errorMessage.style.color = '#f
f6699';
```

💡 **Suggestion:**

Use Canva's Alert component or functional colors like colorDangerFore on colorDangerBack.

Form input has poor visibility HIGH

Light gray text in input field is barely readable, creating form usability issues.

🔍 **Issue Found Here:**

```
input.style.color = '#cccccc'; input.style.backgroundColor = '#f5f5f5';
```

💡 **Suggestion:**

Use Canva's TextInput component with proper focus states and readable text colors.

Medium Priority Issues (6)

Direct DOM manipulation without input sanitization MEDIUM

Using textContent with dynamic data without validation could lead to content injection if user input is later incorporated

🔍 **Issue Found Here:**

```
title.textContent = 'Design Tool Dashboard';
```

💡 **Suggestion:**

Validate and sanitize any dynamic content before setting textContent, even for static strings

Magic numbers and hardcoded values

MEDIUM

Multiple hardcoded values for padding, margins, font sizes without named constants

🔍 **Issue Found Here:**

```
container.style.padding = '16px';
```

💡 **Suggestion:**

Define spacing and typography constants at module level

Inefficient DOM manipulation

MEDIUM

Creating and styling elements individually instead of using document fragments or templates

🔍 **Issue Found Here:**

```
const title = document.createElement('div');
```

💡 **Suggestion:**

Use DocumentFragment for batch DOM operations or template-based approach

Missing error handling for DOM operations

MEDIUM

No error handling for element creation, event listeners, or DOM mounting operations

🔍 **Issue Found Here:**

```
appRoot.appendChild(container);
```

💡 **Suggestion:**

Add try-catch blocks around DOM operations and provide fallback behavior

Non-semantic typography implementation

MEDIUM

Custom font styling instead of Canva's typography system breaks consistency and accessibility.

🔍 **Issue Found Here:**

```
title.style.fontSize = '18px'; title.style.fontFamily = 'Arial, sans-serif';
```

💡 **Suggestion:**

Use Canva's Title component with appropriate size variants (small, medium, large).

Missing proper component structure

MEDIUM

Manual DOM creation bypasses Canva's layout system and responsive design patterns.

🔍 **Issue Found Here:**

```
const container = document.createElement('div');
```

💡 **Suggestion:**

Structure app using Canva's Rows, Columns, and Box components for proper layout and spacing.

Low Priority Issues (4)

Alert usage may expose sensitive information

LOW

Using alert() can potentially expose application state or be used for social engineering attacks

🔍 **Issue Found Here:**

```
alert('Add Element clicked - but this button is hard to see!');
```

💡 **Suggestion:**

Replace alert() with proper UI notifications or logging mechanisms

Unsafe DOM element creation pattern

LOW

Creating elements and setting properties without validation could be exploited if extended with user input

🔍 **Issue Found Here:**

```
const title = document.createElement('h1');
```

💡 **Suggestion:**

Implement input validation wrapper functions for DOM element creation and property setting

Inconsistent element creation pattern

LOW

Mixed approaches to element creation and configuration - some set properties, others use methods

🔍 **Issue Found Here:**

```
input.type = 'text'; vs title.textContent = '...'
```

💡 **Suggestion:**

Standardize element creation with consistent helper function or pattern

Poor separation of data and presentation

LOW

Text content, styles, and structure are mixed together making content updates difficult

🔍 **Issue Found Here:**

```
title.textContent = 'Design Tool Dashboard';
```

💡 **Suggestion:**

Extract text content and style definitions to separate configuration objects

Security Issues (3)

Direct DOM manipulation without input sanitization

MEDIUM

Using textContent with dynamic data without validation could lead to content injection if user input is later incorporated

🔍 **Issue Found Here:**

```
title.textContent = 'Design Tool Dashboard';
```

💡 **Suggestion:**

Validate and sanitize any dynamic content before setting textContent, even for static strings

Alert usage may expose sensitive information

LOW

Using alert() can potentially expose application state or be used for social engineering attacks

🔍 **Issue Found Here:**

```
alert('Add Element clicked - but this button is hard to see!');
```

💡 **Suggestion:**

Replace alert() with proper UI notifications or logging mechanisms

Unsafe DOM element creation pattern

LOW

Creating elements and setting properties without validation could be exploited if extended with user input

🔍 **Issue Found Here:**

```
const title = document.createElement('h1');
```

💡 **Suggestion:**

Implement input validation wrapper functions for DOM element creation and property setting

Code Quality Issues (7)

Monolithic function violates single responsibility HIGH

createPoorContrastApp() handles UI creation, styling, event handling, and mounting - should be split into smaller, focused functions

🔍 **Issue Found Here:**

```
function createPoorContrastApp() {
```

💡 **Suggestion:**

Split into separate functions: createTitle(), createButtons(), createSettings(), mountApp()

Extensive code duplication in styling HIGH

Repeated inline styling patterns throughout the code - no reusable style utilities or constants

🔍 **Issue Found Here:**

```
style.backgroundColor = '#...'; style.color = '#...';
```

💡 **Suggestion:**

Create style constants object and reusable styling functions

Magic numbers and hardcoded values

MEDIUM

Multiple hardcoded values for padding, margins, font sizes without named constants

🔍 **Issue Found Here:**

```
container.style.padding = '16px';
```

💡 **Suggestion:**

Define spacing and typography constants at module level

Inefficient DOM manipulation

MEDIUM

Creating and styling elements individually instead of using document fragments or templates

🔍 **Issue Found Here:**

```
const title = document.createElement('div');
```

💡 **Suggestion:**

Use DocumentFragment for batch DOM operations or template-based approach

Missing error handling for DOM operations

MEDIUM

No error handling for element creation, event listeners, or DOM mounting operations

🔍 **Issue Found Here:**

```
appRoot.appendChild(container);
```

💡 **Suggestion:**

Add try-catch blocks around DOM operations and provide fallback behavior

Inconsistent element creation pattern

LOW

Mixed approaches to element creation and configuration - some set properties, others use methods

🔍 **Issue Found Here:**

```
input.type = 'text'; vs title.textContent = '...'
```

💡 **Suggestion:**

Standardize element creation with consistent helper function or pattern

Poor separation of data and presentation

LOW

Text content, styles, and structure are mixed together making content updates difficult

🔍 **Issue Found Here:**

```
title.textContent = 'Design Tool Dashboard';
```

💡 **Suggestion:**

Extract text content and style definitions to separate configuration objects

UI & UX Issues (8)

Multiple WCAG contrast failures throughout app

CRITICAL

Light gray text on light backgrounds fails WCAG 2.1 AA standards, making content unreadable for users with visual impairments.

🔍 **Issue Found Here:**

```
title.style.color = '#cccccc'; title.style.backgroundColor = '#f0f0f0';
```

💡 **Suggestion:**

Replace custom colors with Canva's functional color tokens like colorTypographyPrimary for proper contrast ratios.

Custom styling violates Canva design system

CRITICAL

Direct style manipulation bypasses Canva's App UI Kit components and design tokens, breaking visual consistency.

🔍 **Issue Found Here:**

```
container.style.backgroundColor = '#f5f5f5';
```

💡 **Suggestion:**

Use Canva's Box, Text, Title, and Button components with functional colors instead of custom CSS.

Yellow text on white background nearly invisible

HIGH

Description text uses light yellow on white, creating severe readability issues visible in the screenshot.

🔍 **Issue Found Here:**

```
description.style.color = '#ffff99'; description.style.backgroundColor = 'whit
e';
```

💡 **Suggestion:**

Use Canva's Text component with colorTypographyPrimary or colorTypographySecondary.

Poor button contrast creates usability barriers

HIGH

Light blue and yellow buttons with light text are barely visible, impacting critical user interactions.

🔍 **Issue Found Here:**

```
addButton.style.backgroundColor = '#b3d9ff'; addButton.style.color = '#f0f0f
0';
```

💡 **Suggestion:**

Replace with Canva's Button component using primary and secondary variants for proper contrast.

Error message has insufficient contrast

HIGH

Red text on pink background fails accessibility standards and doesn't use Canva's error messaging patterns.

🔍 **Issue Found Here:**

```
errorMessage.style.backgroundColor = '#ffccdd'; errorMessage.style.color = '#f
f6699';
```

💡 **Suggestion:**

Use Canva's Alert component or functional colors like colorDangerFore on colorDangerBack.

Form input has poor visibility

HIGH

Light gray text in input field is barely readable, creating form usability issues.

🔍 **Issue Found Here:**

```
input.style.color = '#cccccc'; input.style.backgroundColor = '#f5f5f5';
```

💡 **Suggestion:**

Use Canva's TextInput component with proper focus states and readable text colors.

Non-semantic typography implementation

MEDIUM

Custom font styling instead of Canva's typography system breaks consistency and accessibility.

🔍 **Issue Found Here:**

```
title.style.fontSize = '18px'; title.style.fontFamily = 'Arial, sans-serif';
```

💡 **Suggestion:**

Use Canva's Title component with appropriate size variants (small, medium, large).

Missing proper component structure

MEDIUM

Manual DOM creation bypasses Canva's layout system and responsive design patterns.

🔍 Issue Found Here:

```
const container = document.createElement('div');
```

💡 Suggestion:

Structure app using Canva's Rows, Columns, and Box components for proper layout and spacing.
