Botfellows Design System

Technical Documentation

Version 1.0

Bootstrap Framework

May 29, 2025

Contents

1 Introduction

The ControlsTheme CSS Framework is a comprehensive styling system designed specifically for the Botfellows robot control interface. This framework provides a consistent, maintainable, and scalable approach to styling web applications across all development teams.

1.1 Key Features

- Consistent naming convention with `controls-` prefix
- Brand-aligned color system
- Comprehensive component library
- Mobile-first responsive design
- Zero external dependencies
- Full Bootstrap-like API compatibility

1.2 Design Principles

1. **Namespace Isolation**: All classes are prefixed to prevent conflicts
2. **Semantic Structure**: Classes follow logical naming patterns
3. **Performance Optimized**: Minimal CSS footprint with efficient selectors
4. **Accessibility Compliant**: WCAG 2.1 AA standard compliance

2 Getting Started

2.1 Installation

Include the ControlsTheme CSS file in your project using one of the following methods:

```html
<!-- In your HTML document head -->
<link rel="stylesheet" href="/path/to/ControlsTheme.css">
```

Listing 1: HTML Installation

```javascript
// For React, Vue, or other modern frameworks
import '@/styles/ControlsTheme.css';
```

Listing 2: JavaScript Module Import

2.2 Basic Structure

Every ControlsTheme implementation should begin with the following structure:

```html
<div class="controls">
  <div class="controls-container">
    <!-- Your application content -->
  </div>
</div>
```

Listing 3: Basic HTML Structure

3 Core Concepts

3.1 Naming Convention

The framework employs a consistent naming scheme where all classes are prefixed with `.controls-` to ensure namespace isolation.

Standard CSS/Bootstrap	**ControlsTheme Equivalent**
`.btn`	`.controls-btn`
`.btn-primary`	`.controls-btn-primary`
`.container`	`.controls-container`
`.card`	`.controls-card`

Table 1: Naming Convention Examples

3.2 Color System

The framework utilizes CSS custom properties for color management, enabling consistent theming across all components.

Variable	**Description**	**Value**
`-controls-brand-teal`	Primary brand color	#00a99d
`-controls-brand-dark-teal`	Primary brand color (dark)	#008077
`-controls-brand-black`	Navigation background	#000000
`-controls-primary`	Primary action color	#00a99d
`-controls-danger`	Destructive actions	#dc3545
`-controls-success`	Success states	#00a99d

Table 2: Core Color Variables

4 Component Reference

4.1 Buttons

The button component provides various styles and states for user interactions.

4.1.1 Basic Usage

```html
<!-- Primary Button -->
<button class="controls-btn controls-btn-primary">
  Primary Action
</button>

<!-- Secondary Button -->
<button class="controls-btn controls-btn-secondary">
  Secondary Action
</button>

<!-- Danger Button -->
<button class="controls-btn controls-btn-danger">
  Delete
</button>

<!-- Success Button -->
<button class="controls-btn controls-btn-success">
  Save Changes
</button>
```

Listing 4: Button Variants

4.1.2 Button Sizes

```html
<!-- Small Button -->
<button class="controls-btn controls-btn-sm controls-btn-primary">
  Small
</button>

<!-- Regular Button (default) -->
<button class="controls-btn controls-btn-primary">
  Regular
</button>

<!-- Large Button -->
<button class="controls-btn controls-btn-lg controls-btn-primary">
  Large
</button>

<!-- Block Level Button -->
<button class="controls-btn controls-btn-block controls-btn-primary">
  Full Width Button
</button>
```

Listing 5: Button Size Modifiers

4.1.3 Button Groups

```html
<div class="controls-btn-group">
  <button class="controls-btn controls-btn-primary">First</button>
  <button class="controls-btn controls-btn-primary">Second</button>
  <button class="controls-btn controls-btn-primary">Third</button>
```

```
</div>
```

Listing 6: Button Group Implementation

4.2 Forms

The form components provide consistent styling for user input elements.

4.2.1 Form Structure

```
<form class="controls">
  <div class="controls-form-group">
    <label class="controls-form-label" for="email-input">
      Email Address
    </label>
    <input
      type="email"
      class="controls-form-control"
      id="email-input"
      placeholder="user@example.com"
    >
  </div>

  <div class="controls-form-group">
    <label class="controls-form-label" for="category-select">
      Category
    </label>
    <select class="controls-form-select" id="category-select">
      <option value="">Choose category...</option>
      <option value="1">Manufacturing</option>
      <option value="2">Assembly</option>
      <option value="3">Quality Control</option>
    </select>
  </div>

  <div class="controls-form-group">
    <label class="controls-form-label">
      <input type="checkbox" class="controls-form-checkbox">
      Accept terms and conditions
    </label>
  </div>

  <button type="submit" class="controls-btn controls-btn-primary">
    Submit Form
  </button>
</form>
```

Listing 7: Complete Form Example

4.2.2 Input Groups

```
<div class="controls-input-group">
  <input
    type="text"
```

```html
    class="controls-form-control"
    placeholder="Search robots..."
  >
  <button class="controls-btn controls-btn-primary">
    Search
  </button>
</div>
```

Listing 8: Input Group with Button

4.3 Cards

Cards provide a flexible content container with multiple variants.

4.3.1 Basic Card

```html
<div class="controls-card">
  <div class="controls-card-body">
    <h5 class="controls-card-title">Card Title</h5>
    <h6 class="controls-card-subtitle">Card Subtitle</h6>
    <p class="controls-card-text">
      This is the card's main content area where you can
      place any relevant information.
    </p>
    <button class="controls-btn controls-btn-primary">
      Action
    </button>
  </div>
</div>
```

Listing 9: Standard Card Component

4.3.2 Card with Header and Footer

```html
<div class="controls-card">
  <div class="controls-card-header">
    Featured Content
  </div>
  <div class="controls-card-body">
    <h5 class="controls-card-title">Special Title</h5>
    <p class="controls-card-text">
      With supporting text below as a natural lead-in.
    </p>
  </div>
  <div class="controls-card-footer">
    Last updated 3 mins ago
  </div>
</div>
```

Listing 10: Complete Card Structure

4.4 Grid System

The grid system provides flexible layout options using CSS Grid.

4.4.1 Basic Grid

```html
<!-- Three Column Grid -->
<div class="controls-grid controls-grid-cols-3 controls-gap-md">
  <div class="controls-card">Column 1</div>
  <div class="controls-card">Column 2</div>
  <div class="controls-card">Column 3</div>
</div>

<!-- Four Column Grid with Small Gap -->
<div class="controls-grid controls-grid-cols-4 controls-gap-sm">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
  <div>Item 4</div>
</div>
```

Listing 11: Grid Layout Examples

4.5 Navigation

The navigation component provides a consistent header structure.

```html
<nav class="controls-navbar controls-navbar-dark controls-bg-primary"
   >
  <div class="controls-container-fluid">
    <a class="controls-navbar-brand" href="#">
      <img src="/logo.png" alt="Botfellows" style="height: 36px;">
    </a>

    <div class="controls-navbar-nav controls-ms-auto">
      <a class="controls-nav-link active" href="#">Workspace</a>
      <a class="controls-nav-link" href="#">Robot</a>
      <a class="controls-nav-link" href="#">Simulation</a>
      <a class="controls-nav-link" href="#">Analytics</a>
      <a class="controls-nav-link" href="#">Documentation</a>
    </div>
  </div>
</nav>
```

Listing 12: Navigation Bar Implementation

5 Utility Classes

5.1 Spacing Utilities

The framework provides comprehensive spacing utilities following a consistent scale.

Class	Property	Description
`.controls-m-0`	margin: 0	Remove all margins
`.controls-m-1`	margin: 0.25rem	Small margin
`.controls-m-2`	margin: 0.5rem	Medium margin
`.controls-m-3`	margin: 1rem	Default margin
`.controls-m-4`	margin: 1.5rem	Large margin
`.controls-m-5`	margin: 3rem	Extra large margin

Table 3: Margin Utilities

Class	Property	Description
`.controls-p-0`	padding: 0	Remove all padding
`.controls-p-1`	padding: 0.25rem	Small padding
`.controls-p-2`	padding: 0.5rem	Medium padding
`.controls-p-3`	padding: 1rem	Default padding
`.controls-p-4`	padding: 1.5rem	Large padding
`.controls-p-5`	padding: 3rem	Extra large padding

Table 4: Padding Utilities

5.2 Display Utilities

```html
<!-- Hide element -->
<div class="controls-d-none">Hidden content</div>

<!-- Block display -->
<span class="controls-d-block">Block element</span>

<!-- Inline block -->
<div class="controls-d-inline-block">Inline block</div>

<!-- Flexbox container -->
<div class="controls-d-flex">Flex container</div>
```

Listing 13: Display Property Classes

5.3 Flexbox Utilities

```html
<!-- Horizontal alignment -->
<div class="controls-d-flex controls-justify-content-between">
  <span>Left</span>
  <span>Right</span>
</div>

<!-- Vertical alignment -->
<div class="controls-d-flex controls-align-items-center">
  <div>Vertically centered content</div>
</div>

```

```html
<!-- Combined alignment -->
<div class="controls-d-flex
            controls-justify-content-center
            controls-align-items-center">
  <div>Perfectly centered</div>
</div>
```

Listing 14: Flexbox Layout Utilities

5.4 Text Utilities

Class	Description
`.controls-text-primary`	Primary brand color text
`.controls-text-secondary`	Secondary color text
`.controls-text-success`	Success state text
`.controls-text-danger`	Error/danger state text
`.controls-text-muted`	Muted/disabled text
`.controls-text-center`	Center aligned text
`.controls-text-right`	Right aligned text
`.controls-font-weight-bold`	Bold text weight

Table 5: Text Utility Classes

6 Migration Guide

6.1 Migration from Bootstrap

For teams currently using Bootstrap, the following table provides a comprehensive mapping:

Bootstrap Class	ControlsTheme Class
`.btn`	`.controls-btn`
`.btn-primary`	`.controls-btn-primary`
`.btn-secondary`	`.controls-btn-secondary`
`.btn-danger`	`.controls-btn-danger`
`.btn-success`	`.controls-btn-success`
`.btn-sm`	`.controls-btn-sm`
`.btn-lg`	`.controls-btn-lg`
`.form-control`	`.controls-form-control`
`.form-group`	`.controls-form-group`
`.form-label`	`.controls-form-label`
`.card`	`.controls-card`
`.card-body`	`.controls-card-body`
`.card-title`	`.controls-card-title`
`.alert`	`.controls-alert`
`.alert-danger`	`.controls-alert-danger`
`.badge`	`.controls-badge`
`.container`	`.controls-container`
`.row`	`.controls-grid`
`.mt-3`	`.controls-mt-3`
`.mb-3`	`.controls-mb-3`
`.p-3`	`.controls-p-3`
`.d-flex`	`.controls-d-flex`
`.d-none`	`.controls-d-none`

Table 6: Bootstrap to ControlsTheme Migration Map

6.2 Automated Migration

For large codebases, use the following migration script:

Listing 15: Automated Migration Script

```
// migration-script.js
const fs = require('fs');
const path = require('path');

const migrationMap = {
  'class="btn': 'class="controls-btn',
  'class="form-': 'class="controls-form-',
  'class="card': 'class="controls-card',
  'class="alert': 'class="controls-alert',
  'class="badge': 'class="controls-badge',
  'class="container': 'class="controls-container',
  'class="mt-': 'class="controls-mt-',
  'class="mb-': 'class="controls-mb-',
  'class="p-': 'class="controls-p-',
  'class="d-': 'class="controls-d-',
  'className="btn': 'className="controls-btn',
  'className="form-': 'className="controls-form-',
  'className="card': 'className="controls-card'
};
```

```
function migrateFile(filePath) {
  let content = fs.readFileSync(filePath, 'utf8');

  Object.entries(migrationMap).forEach(([oldPattern, newPattern]) => {
    const regex = new RegExp(oldPattern.replace(/[.*+?^${}()|[\]\\]/g, '\\$&'),
    content = content.replace(regex, newPattern);
  });

  fs.writeFileSync(filePath, content);
  console.log(`Migrated: ${filePath}`);
}

// Usage: node migration-script.js ./src
const targetDir = process.argv[2] || './src';
// Implement recursive file processing...
```

7 Best Practices

7.1 Structural Guidelines

1. **Root Container**: Always wrap your application content in a `.controls` container
2. **Semantic HTML**: Use appropriate HTML elements for their intended purpose
3. **Consistent Spacing**: Utilize the spacing utility classes rather than inline styles
4. **Component Composition**: Build complex interfaces by composing simple components

7.2 Performance Optimization

1. **Minimize Overrides**: Use framework classes directly rather than creating custom CSS
2. **Leverage CSS Variables**: Customize theme colors using CSS custom properties
3. **Avoid Deep Nesting**: Keep HTML structure shallow for better performance
4. **Use Utility Classes**: Prefer utility classes over component-specific styles

7.3 Accessibility Considerations

1. **Semantic Markup**: Use proper heading hierarchy and ARIA labels
2. **Focus States**: All interactive elements include visible focus indicators
3. **Color Contrast**: Maintain WCAG 2.1 AA compliance for text contrast
4. **Keyboard Navigation**: Ensure all functionality is keyboard accessible

8 Common Patterns

8.1 Robot Control Panel

```html
<div class="controls">
  <div class="controls-section">
    <div class="controls-section-header">
      <h3 class="controls-section-title">Joint Controls</h3>
      <button class="controls-btn controls-btn-primary">
        Reset All
      </button>
    </div>

    <div class="controls-card-body">
      <div class="controls-form-group">
        <label class="controls-form-label">Joint 1 Position</label>
        <div class="controls-d-flex controls-align-items-center">
          <input
            type="range"
            class="controls-form-control"
            min="-180"
            max="180"
          >
          <span class="controls-ml-3">0  </span>
        </div>
      </div>

      <div class="controls-form-group">
        <label class="controls-form-label">Joint 2 Position</label>
        <div class="controls-d-flex controls-align-items-center">
          <input
            type="range"
            class="controls-form-control"
            min="-180"
            max="180"
          >
          <span class="controls-ml-3">0  </span>
        </div>
      </div>
    </div>
  </div>
</div>
```

Listing 16: Robot Control Panel Implementation

8.2 Status Dashboard

```html
<div class="controls">
  <div class="controls-grid controls-grid-cols-4 controls-gap-md">
    <!-- Status Card 1 -->
    <div class="controls-card">
      <div class="controls-card-body controls-text-center">
        <h5 class="controls-h5">Active Robots</h5>
        <p class="controls-h2 controls-text-primary">12</p>
        <span class="controls-badge controls-badge-success">
          Online
        </span>
      </div>
    </div>

    <!-- Status Card 2 -->
    <div class="controls-card">
      <div class="controls-card-body controls-text-center">
        <h5 class="controls-h5">Tasks Completed</h5>
        <p class="controls-h2 controls-text-primary">1,247</p>
        <span class="controls-text-muted">Today</span>
      </div>
    </div>

    <!-- Status Card 3 -->
    <div class="controls-card">
      <div class="controls-card-body controls-text-center">
        <h5 class="controls-h5">Error Rate</h5>
        <p class="controls-h2 controls-text-success">0.3%</p>
        <span class="controls-text-muted">Last 24h</span>
      </div>
    </div>

    <!-- Status Card 4 -->
    <div class="controls-card">
      <div class="controls-card-body controls-text-center">
        <h5 class="controls-h5">Efficiency</h5>
        <p class="controls-h2 controls-text-primary">94.5%</p>
        <span class="controls-badge controls-badge-primary">
          Optimal
        </span>
      </div>
    </div>
  </div>
</div>
```

Listing 17: Dashboard Layout Pattern

9 Customization

9.1 Theme Customization

Override CSS variables to customize the theme:

```
/* custom-theme.css */
:root {
  /* Override brand colors */
  --controls-brand-teal: #00b8a9;
  --controls-brand-dark-teal: #008f82;

  /* Adjust spacing scale */
  --controls-spacing-1: 0.375rem;
  --controls-spacing-2: 0.75rem;
  --controls-spacing-3: 1.5rem;
  --controls-spacing-4: 2.25rem;
  --controls-spacing-5: 4.5rem;

  /* Modify border radius */
  --controls-border-radius: 0.5rem;
  --controls-border-radius-sm: 0.375rem;
  --controls-border-radius-lg: 0.75rem;

  /* Adjust font settings */
  --controls-font-family: 'Inter', -apple-system, sans-serif;
  --controls-font-size-base: 0.9375rem;

  /* Customize shadows */
  --controls-box-shadow: 0 0.5rem 2rem rgba(0,0,0,0.1);
  --controls-box-shadow-lg: 0 1rem 4rem rgba(0,0,0,0.15);
}
```

Listing 18: Custom Theme Variables

9.2 Component Extension

Create custom components while maintaining consistency:

```
/* Custom status indicator component */
.controls-status-indicator {
  display: inline-flex;
  align-items: center;
  padding: var(--controls-spacing-1) var(--controls-spacing-2);
  border-radius: var(--controls-border-radius-sm);
  font-size: var(--controls-font-size-sm);
  font-weight: 500;
}

.controls-status-indicator::before {
  content: '';
  display: inline-block;
  width: 8px;
  height: 8px;
  border-radius: 50%;
  margin-right: var(--controls-spacing-1);
  background-color: currentColor;
}

.controls-status-indicator-active {
  color: var(--controls-success);
  background-color: rgba(0, 169, 157, 0.1);
}

.controls-status-indicator-inactive {
  color: var(--controls-gray-500);
  background-color: var(--controls-gray-100);
}

.controls-status-indicator-error {
  color: var(--controls-danger);
  background-color: rgba(220, 53, 69, 0.1);
}
```

Listing 19: Custom Component Example

10 Troubleshooting

10.1 Common Issues

Issue	**Solution**
Styles not applying	Ensure the `.controls` wrapper is present and CSS file is properly loaded
Class conflicts	Verify all classes use the `controls-` prefix
Responsive issues	Check viewport meta tag and use appropriate grid classes
Color inconsistency	Use CSS variables instead of hard-coded color values
Missing components	Verify you're using the latest version of ControlsTheme.css

Table 7: Common Issues and Solutions

10.2 Browser Compatibility

The framework supports the following browsers:

- Chrome/Edge: Latest 2 versions
- Firefox: Latest 2 versions
- Safari: Latest 2 versions
- Mobile browsers: iOS Safari 12+, Chrome Mobile

11 Appendices

11.1 Appendix A: Complete Class Reference

A comprehensive list of all available classes is maintained in the framework source file. Key categories include:

- **Layout**: Container, grid, flexbox utilities
- **Components**: Buttons, cards, forms, navigation
- **Utilities**: Spacing, display, text, color
- **Helpers**: Borders, shadows, positioning

11.2 Appendix B: CSS Variable Reference

All customizable CSS variables are documented in the framework source with their default values and usage examples.

11.3 Appendix C: Version History

Version	**Date**	**Changes**
1.0.0	2024-01	Initial release with complete component library

Table 8: Version History