DESIGN SYSTEM · COLOUR

Colour System Usage Guide

Semantic colour tokens, primitive palettes, and implementation patterns for building consistent interfaces.

59 semantic tokens · 7 primitive palettes · Tokens Studio compatible

Overview

The colour system is structured in two layers. Primitives define the raw palette; semantic tokens assign purpose. Always use semantic tokens in components.

Rule: Never reference a primitive token directly in a component. Always use a semantic token — e.g. `var(--color-text-primary)` not `var(--color-greyscale-100)`.

Architecture

Primitives

Raw colour values — the source palette. 7 scales (Red, Orange, Yellow, Green, Blue, Greyscale, Black & White), each with 10 steps.

Semantic

Purpose-driven aliases of primitives. 59 tokens across Background, Text, Border, Icon, Focus, and Input groups.

Primitive Palettes

GREEN · BRAND PALETTE

Green 10	`#f3fffc`
Green 20	`#d0fff6`
Green 30	`#acffef`
Green 40	`#85f6e0`
Green 50	`#6cd9c4`
Green 60	`#56bca8`
Green 70	`#429f8d`
Green 80	`#318272`
Green 90	`#226558`
Green 100	`#15483e`

BLUE · INFORMATIONAL

Blue 10	`#ecf7ff`
Blue 20	`#b7e0ff`
Blue 30	`#8accff`
Blue 40	`#5cb8ff`
Blue 50	`#2ea4ff`
Blue 60	`#0088ef`
Blue 70	`#0070c6`
Blue 80	`#00599d`
Blue 90	`#004275`
Blue 100	`#002b4c`

RED · ERROR / DESTRUCTIVE

Red 10	`#ffeae7`
Red 20	`#ffc5bd`
Red 30	`#ffa193`
Red 40	`#ff7c69`
Red 50	`#ff573f`
Red 60	`#ff3215`
Red 70	`#e51c00`
Red 80	`#bc1700`
Red 90	`#931200`

Red 100	#6b0d00

YELLOW · WARNING

Yellow 10	#fffbed
Yellow 20	#fff3c4
Yellow 30	#ffeb9b
Yellow 40	#ffe273
Yellow 50	#f2cf46
Yellow 60	#d0b032
Yellow 70	#ae9121
Yellow 80	#8c7314
Yellow 90	#6a560a
Yellow 100	#483a03

GREYSCALE · NEUTRAL

Grey 10	#f5f5f5
Grey 20	#e1e1e1
Grey 30	#cccccc
Grey 40	#b8b8b8
Grey 50	#a3a3a3
Grey 60	#8f8f8f
Grey 70	#7b7b7b
Grey 80	#656566
Grey 90	#4f5052
Grey 100	#3b3b3d

Semantic Tokens — Background

Use these tokens for all container, surface, and page-level backgrounds.

BASE

	TOKEN	HEX	USAGE
⬜	`Background.Page`	`#ffffff`	Main page / document background
⬜	`Background.Surface`	`#ffffff`	Card, modal, popover surfaces
⬜	`Background.Subtle`	`#f5f5f5`	Subdued sections, zebra rows
⬜	`Background.Muted`	`#e1e1e1`	Skeleton loaders, fills
⬛	`Background.Inverse`	`#3b3b3d`	Dark tooltips, inverse sections

BRAND

	TOKEN	HEX	USAGE
🟩	`Background.Brand.Primary`	`#318272`	Primary CTA button, brand sections
🟩	`Background.Brand.Hover`	`#226558`	CTA button hover state
🟩	`Background.Brand.Pressed`	`#15483e`	CTA button active / pressed state
⬜	`Background.Brand.Subtle`	`#f3fffc`	Tinted brand highlights, banners

STATE

	TOKEN	HEX	USAGE
⬜	`Background.State.Success.Subtle`	`#f3fffc`	Success alert background
🟩	`Background.State.Success.Bold`	`#318272`	Filled success badge
⬜	`Background.State.Info.Subtle`	`#ecf7ff`	Info alert / banner background
🟦	`Background.State.Info.Bold`	`#00599d`	Filled info badge
⬜	`Background.State.Warning.Subtle`	`#fffbed`	Warning alert background
🟨	`Background.State.Warning.Bold`	`#8c7314`	Filled warning badge
⬜	`Background.State.Error.Subtle`	`#ffeae7`	Error alert background
🟥	`Background.State.Error.Bold`	`#bc1700`	Filled error badge, destructive button

Semantic Tokens — Text

Use these tokens for all text elements. Never hard-code a hex value for text colour.

BASE

	TOKEN	HEX	USAGE
⬛	`Text.Primary`	`#3b3b3d`	Body text, headings — default
⬛	`Text.Secondary`	`#4f5052`	Supporting labels, captions
⬛	`Text.Tertiary`	`#7b7b7b`	Helper text, metadata
⬛	`Text.Placeholder`	`#8f8f8f`	Input placeholder text
⬛	`Text.Disabled`	`#8f8f8f`	Disabled field labels
⬜	`Text.Inverse`	`#ffffff`	Text on dark / brand backgrounds
🟩	`Text.Brand`	`#226558`	Brand links, active nav items

STATE

	TOKEN	HEX	USAGE
🟩	`Text.State.Success`	`#226558`	Success message text
🟦	`Text.State.Info`	`#004275`	Info message text
🟫	`Text.State.Warning`	`#6a560a`	Warning message text
🟥	`Text.State.Error`	`#931200`	Error / validation message text

Semantic Tokens — Border

Controls strokes, dividers, and outlines on containers and interactive elements.

BASE

	TOKEN	HEX	USAGE
⬜	`Border.Default`	`#cccccc`	Default input, card borders
⬜	`Border.Subtle`	`#e1e1e1`	Hairline dividers, separators
◼	`Border.Strong`	`#a3a3a3`	Prominent dividers, table borders
⬜	`Border.Disabled`	`#e1e1e1`	Disabled input borders
⬛	`Border.Inverse`	`#3b3b3d`	High contrast borders on light bg
🟩	`Border.Brand`	`#318272`	Focused / selected borders

STATE

	TOKEN	HEX	USAGE
🟩	`Border.State.Success`	`#318272`	Success-state container border
🟦	`Border.State.Info`	`#00599d`	Info-state border
🟨	`Border.State.Warning`	`#8c7314`	Warning-state border
🟥	`Border.State.Error`	`#bc1700`	Error-state input border

Semantic Tokens — Icon

Controls icon fills. Pair with the matching text token for consistent hierarchy.

	TOKEN	HEX	USAGE
⬛	`Icon.Primary`	`#4f5052`	Default icons beside primary text
⬛	`Icon.Secondary`	`#7b7b7b`	Supporting / decorative icons
⬛	`Icon.Disabled`	`#8f8f8f`	Icons in disabled components
⬜	`Icon.Inverse`	`#ffffff`	Icons on dark / brand backgrounds
🟩	`Icon.Brand`	`#226558`	Brand icons, active states
🟩	`Icon.State.Success`	`#226558`	Success icon
🟦	`Icon.State.Info`	`#004275`	Info icon
🟫	`Icon.State.Warning`	`#6a560a`	Warning icon
🟥	`Icon.State.Error`	`#931200`	Error icon

Semantic Tokens — Focus

Focus rings ensure keyboard-navigable interfaces meet WCAG 2.1 AA (3:1 contrast against adjacent colour).

	TOKEN	HEX · USAGE
🟦	`Focus.Ring`	#acffef · Standard focus ring on all interactive elements
🟥	`Focus.RingError`	#ffa193 · Focus ring when element is in error state

Semantic Tokens — Input

Scoped specifically to form input components. Covers all five interactive states.

BACKGROUND

	TOKEN	HEX · USAGE
⬜	`Input.Background.Default`	#ffffff · Default, hover, focus, error
⬜	`Input.Background.Disabled`	#e1e1e1 · Disabled input background

BORDER

	TOKEN	HEX · USAGE
⬜	Input.Border.Default	#cccccc · Default (unfocused) border
⬜	Input.Border.Hover	#a3a3a3 · Mouse hover border
🟩	Input.Border.Focus	#318272 · Focused border
🟥	Input.Border.Error	#bc1700 · Error state border
⬜	Input.Border.Disabled	#e1e1e1 · Disabled border

TEXT

	TOKEN	HEX · USAGE
⬛	Input.Text.Value	#3b3b3d · Typed / selected value
⬜	Input.Text.Placeholder	#8f8f8f · Placeholder text
⬜	Input.Text.Disabled	#8f8f8f · Text in disabled inputs

Implementation Examples

Practical CSS patterns using the semantic token CSS variables. Generate variables from `tokens/tokens.updated.json` using Style Dictionary.

CSS Custom Properties

```css
:root {
  /* Background */
  --color-bg-page:             #ffffff;
  --color-bg-surface:          #ffffff;
  --color-bg-subtle:           #f5f5f5;
  --color-bg-muted:            #e1e1e1;
  --color-bg-inverse:          #3b3b3d;
  --color-bg-brand-primary:    #318272;
  --color-bg-brand-hover:      #226558;
  --color-bg-brand-pressed:    #15483e;

  /* Text */
  --color-text-primary:        #3b3b3d;
  --color-text-secondary:      #4f5052;
  --color-text-tertiary:       #7b7b7b;
  --color-text-placeholder:    #8f8f8f;
  --color-text-inverse:        #ffffff;
  --color-text-brand:          #226558;

  /* Border */
  --color-border-default:      #cccccc;
  --color-border-subtle:       #e1e1e1;
  --color-border-strong:       #a3a3a3;
  --color-border-brand:        #318272;
  --color-border-error:        #bc1700;

  /* Focus */
  --color-focus-ring:          #acffef;
  --color-focus-ring-error:    #ffa193;

  /* Input */
  --color-input-border-default:    #cccccc;
  --color-input-border-hover:      #a3a3a3;
  --color-input-border-focus:      #318272;
  --color-input-border-error:      #bc1700;
  --color-input-border-disabled:   #e1e1e1;
  --color-input-bg-disabled:       #e1e1e1;
  --color-input-text-value:        #3b3b3d;
  --color-input-text-placeholder:  #8f8f8f;
}
```

Input Field

```css
.input {
  background-color: var(--color-input-bg-default);
  border: 1px solid var(--color-input-border-default);
  color: var(--color-input-text-value);
}
.input::placeholder { color: var(--color-input-text-placeholder); }

.input:hover       { border-color: var(--color-input-border-hover); }

.input:focus {
  border-color: var(--color-input-border-focus);
  outline: none;
  box-shadow: 0 0 0 3px var(--color-focus-ring);
}
.input--error       { border-color: var(--color-input-border-error); }
.input--error:focus { box-shadow: 0 0 0 3px var(--color-focus-ring-error); }

.input:disabled {
  background-color: var(--color-input-bg-disabled);
  border-color: var(--color-input-border-disabled);
  color: var(--color-input-text-disabled);
  cursor: not-allowed;
}
```

Button

```css
.btn--primary {
  background-color: var(--color-bg-brand-primary);
  color: var(--color-text-inverse);
}
.btn--primary:hover         { background-color: var(--color-bg-brand-hover); }
.btn--primary:active        { background-color: var(--color-bg-brand-pressed); }
.btn--primary:focus-visible { box-shadow: 0 0 0 3px var(--color-focus-ring); }
```

Alert / Notification

```
.alert--success {
  background-color: var(--color-bg-success-subtle);
  border: 1px solid var(--color-border-success);
  color: var(--color-text-success);
}
.alert--info {
  background-color: var(--color-bg-info-subtle);
  border: 1px solid var(--color-border-info);
  color: var(--color-text-info);
}
.alert--warning {
  background-color: var(--color-bg-warning-subtle);
  border: 1px solid var(--color-border-warning);
  color: var(--color-text-warning);
}
.alert--error {
  background-color: var(--color-bg-error-subtle);
  border: 1px solid var(--color-border-error);
  color: var(--color-text-error);
}
```

Dos and Don'ts

DO

- ✓ Use semantic tokens in all component code
- ✓ Pair matching semantic text + background tokens
- ✓ Always provide a visible focus style
- ✓ Use `Focus.Ring` on all interactive elements
- ✓ Verify contrast before pairing subtle tokens

DON'T

- ✕ Reference primitive tokens in components
- ✕ Hard-code hex values in CSS
- ✕ Use `Text.Placeholder` for body text
- ✕ Pair `Subtle` background with `Subtle` text
- ✕ Skip focus styles — accessibility violation