

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

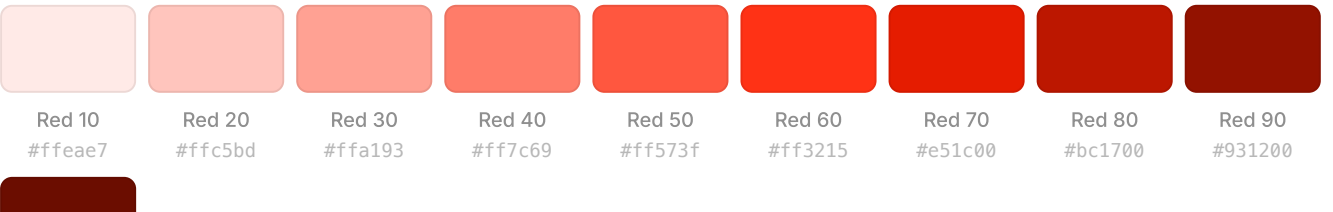
GREEN · BRAND PALETTE



BLUE · INFORMATIONAL



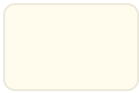
RED · ERROR / DESTRUCTIVE





Red 100
#6b0d00

YELLOW · WARNING



Yellow 10
#ffffbed



Yellow 20
#ffff3c4



Yellow 30
#fffeb9b



Yellow 40
#ffe273



Yellow 50
#f2cf46



Yellow 60
#d0b032



Yellow 70
#ae9121



Yellow 80
#8c7314

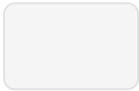


Yellow 90
#6a560a

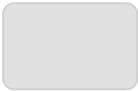


Yellow 100
#483a03

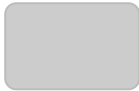
GREYSCALE · NEUTRAL



Grey 10
#f5f5f5



Grey 20
#e1e1e1



Grey 30
#cccccc



Grey 40
#b8b8b8



Grey 50
#a3a3a3



Grey 60
#8f8f8f



Grey 70
#7b7b7b



Grey 80
#656565



Grey 90
#4f5052



Grey 100
#3b3b3d



Semantic Tokens — Background

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









BASE

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

BRAND

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








STATE

TOKEN	HEX	USAGE
 <code>Background.State.Success.Subtle</code>	<code>#f3fffc</code>	Success alert background
 <code>Background.State.Success.Bold</code>	<code>#318272</code>	Filled success badge
 <code>Background.State.Info.Subtle</code>	<code>#ecf7ff</code>	Info alert / banner background
 <code>Background.State.Info.Bold</code>	<code>#00599d</code>	Filled info badge
 <code>Background.State.Warning.Subtle</code>	<code>#fffbed</code>	Warning alert background
 <code>Background.State.Warning.Bold</code>	<code>#8c7314</code>	Filled warning badge
 <code>Background.State.Error.Subtle</code>	<code>#ffeae7</code>	Error alert background
 <code>Background.State.Error.Bold</code>	<code>#bc1700</code>	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
 <code>Text.Primary</code>	#3b3b3d	Body text, headings — default
 <code>Text.Secondary</code>	#4f5052	Supporting labels, captions
 <code>Text.Tertiary</code>	#7b7b7b	Helper text, metadata
 <code>Text.Placeholder</code>	#8f8f8f	Input placeholder text
 <code>Text.Disabled</code>	#8f8f8f	Disabled field labels
 <code>Text.Inverse</code>	#ffffff	Text on dark / brand backgrounds
 <code>Text.Brand</code>	#226558	Brand links, active nav items


STATE

TOKEN	HEX	USAGE
 <code>Text.State.Success</code>	#226558	Success message text
 <code>Text.State.Info</code>	#004275	Info message text
 <code>Text.State.Warning</code>	#6a560a	Warning message text
 <code>Text.State.Error</code>	#931200	Error / validation message text





Semantic Tokens — Border

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

BASE










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

STATE

TOKEN	HEX	USAGE
 <code>Border.State.Success</code>	<code>#318272</code>	Success-state container border
 <code>Border.State.Info</code>	<code>#00599d</code>	Info-state border
 <code>Border.State.Warning</code>	<code>#8c7314</code>	Warning-state border
 <code>Border.State.Error</code>	<code>#bc1700</code>	Error-state input border


Semantic Tokens — Icon

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

TOKEN	HEX	USAGE
 <code>Icon.Primary</code>	<code>#4f5052</code>	Default icons beside primary text
 <code>Icon.Secondary</code>	<code>#7b7b7b</code>	Supporting / decorative icons
 <code>Icon.Disabled</code>	<code>#8f8f8f</code>	Icons in disabled components
 <code>Icon.Inverse</code>	<code>#ffffff</code>	Icons on dark / brand backgrounds
 <code>Icon.Brand</code>	<code>#226558</code>	Brand icons, active states
 <code>Icon.State.Success</code>	<code>#226558</code>	Success icon
 <code>Icon.State.Info</code>	<code>#004275</code>	Info icon
 <code>Icon.State.Warning</code>	<code>#6a560a</code>	Warning icon
 <code>Icon.State.Error</code>	<code>#931200</code>	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
 <code>Focus.Ring</code>	<code>#acffef</code> · Standard focus ring on all interactive elements
 <code>Focus.RingError</code>	<code>#ffa193</code> · 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
 <code>Input.Background.Default</code>	<code>#ffffff</code> · Default, hover, focus, error
 <code>Input.Background.Disabled</code>	<code>#e1e1e1</code> · Disabled input background

BORDER

TOKEN	HEX · USAGE
 <code>Input.Border.Default</code>	<code>#cccccc</code> · Default (unfocused) border
 <code>Input.Border.Hover</code>	<code>#a3a3a3</code> · Mouse hover border
 <code>Input.Border.Focus</code>	<code>#318272</code> · Focused border
 <code>Input.Border.Error</code>	<code>#bc1700</code> · Error state border
 <code>Input.Border.Disabled</code>	<code>#e1e1e1</code> · Disabled border

TEXT

TOKEN	HEX · USAGE
 <code>Input.Text.Value</code>	<code>#3b3b3d</code> · Typed / selected value
 <code>Input.Text.Placeholder</code>	<code>#8f8f8f</code> · Placeholder text
 <code>Input.Text.Disabled</code>	<code>#8f8f8f</code> · 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

```
: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

```
.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

```
.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