Theming

# Theming System

## Introduction

**About this document:** This document explains the theming architecture of the Oblique Design System, including theme types, token organization, and implementation patterns.

**Scope:** Tokenized Design System only. Pre-Design System releases like Oblique R13 are not affected.

**Theme Strategy:** The system supports multiple theme dimensions that can be combined: Lightness (light/dark), Inversity (default/inverse), and Interaction Emphasis (default/muted).

---

## Design Token Architecture

### Token Hierarchy Overview

Design tokens reference each other through a hierarchical structure with three main levels:

1. \*\*Primitives\*\* (Level 1) - Base color values
2. \*\*Semantics\*\* (Level 2) - Contextual color meanings with theming support
3. \*\*Components\*\* (Level 3) - Component-specific token references

**Reference Chain:** `Components → Semantics → Primitives`

The component level references the semantic level, which references the primitive level. Only the primitive level contains hardcoded values. Code and Figma consume semantic and component levels, but we don't expose the component level to design system consumers in the published libraries.

### Multi-Dimensional Theming

The semantic level supports complex theming through multiple dimensions that can be combined:

* \*\*Lightness:\*\* `light` (default) / `dark`
* \*\*Inversity:\*\* `default` / `inverse`
* \*\*Interaction Emphasis:\*\* `default` / `muted`

**Example:** A button can simultaneously use:

* `lightness: dark` (inheriting dark theme values)
* `inversity: inverse` (switching to inverted background/foreground for visual emphasis)
* `emphasis: muted` (lowering visual emphasis within headers/footers)

### Level 1: Primitives

src/lib/themes/primitive/color.json

Contains only hardcoded hex values organized by color groups without semantic meaning:

{

"ob": {

"p": {

"colors": {

"red": { "50": "#fff5f5", "600": "#dc2626", "900": "#7f1d1d" },

"blue": { "50": "#eff6ff", "600": "#2563eb", "900": "#1e3a8a" },

"cobalt": { "50": "#f8fafc", "600": "#475569", "900": "#0f172a" }

}

}

}

}

### Level 2: Semantics

The semantic level introduces theming folders that enable theme switching. Each folder contains tokens with identical names but different values, allowing one to override the other when activated.

#### Level 2.1: Lightness Theming

src/lib/themes/semantics/colors/lightness/

├── light.json # Default theme

└── dark.json # Dark theme override

**Purpose:** System-wide light/dark theme switching. Users set the variable mode in Figma from `lightness: light` (default) or `lightness: dark`.

**Token Structure:** Generic-purpose tokens with contrast levels:

* `bg-base` - For various backgrounds across different interaction states
* `fg-visited` - Reserved for foreground color of visited text links
* `contrast-high`, `contrast-medium`, `contrast-low` - Various contrast levels within semantic subgroups

**Token Format:**

{

"ob": {

"s": {

"color": {

"neutral": {

"bg": { "contrast-highest-default": "{ob.p.color.basic.white}",

"contrast-highest-inverse": "{ob.p.color.cobalt.700}"

}

},

"interaction": {

"emphasis-default": {

"bg-base": { "contrast-high-default": "{ob.p.color.steelblue.800}",

"contrast-high-inverse": "{ob.p.color.basic.white}"

}

}

}

}

}

}

}

#### Level 2.2: Inversity Theming

src/lib/themes/semantics/colors/inversity/

├── default.json # Default contrast

└── inverse.json # Inverted contrast

**Purpose:** Component-scoped theming for inversity override. Available in Figma variable mode but should not be altered by designers in projects.

**Use Cases:**

* Setting complete inversity on components: Badge needs inverted appearance (dark bg, light fg)
* One-time inversity on component variants: `button.primary`, `infobox.fatal`, `tag.active` to make them stick out for UX reasons

**Token Structure:** References lightness tokens without theme suffixes:

{

"ob": {

"s": {

"color": {

"neutral": {

"bg": {

"contrast-highest": "{ob.s.color.neutral.bg.contrast-highest-default}"

}

}

}

}

}

}

#### Level 2.3: Interaction Emphasis Theming

src/lib/themes/semantics/colors/emphasis/

├── default.json # Full saturation (standard blue)

└── muted.json # Desaturated (monochromatic)

**Purpose:** Component-scoped theming for interaction emphasis. Used when design system maintainers set interaction emphasis to "muted" in host components (header, footer, infobox) where text links and buttons must be visually less aggressive.

**Use Cases:**

* \*\*Standard Context:\*\* Buttons in forms use `emphasis: default` with high-saturated blue for background fill
* \*\*Muted Context:\*\* Same components in headers/footers use `emphasis: muted` appearing monochromatic to avoid drawing excessive attention

**Scope:** Only for interactive elements (buttons, text links, tabs). Non-interactive components can skip this level.

**Token Structure:** References inversity tokens:

{

"ob": {

"s": {

"color": {

"interaction": {

"state": {

"fg": {

"enabled-default": "{ob.s.color.status.fatal.fg.contrast-low}",

"hover-default": "{ob.s.color.status.fatal.fg.contrast-medium}"

}

}

}

}

}

}

}

### Level 3: Components

src/lib/themes/html/button/color-static.json

**Purpose:** Component-specific token references. No theming subfolders should exist at this level when possible.

**Reference Strategy:** Components reference inversity tokens and remain agnostic to whether they're inverse, muted, or on dark theme - all theming happens at the semantic level.

**Token Structure:** References semantic tokens pre-defined for specific use cases:

{

"ob": {

"h": {

"button": {

"color": {

"bg": {

"primary": {

"enabled": "{ob.s.color.interaction.state.bg.enabled-inverse}"

}

}

}

}

}

}

}

### Reference Chain Validation

The correct reference chain ensures proper theme inheritance:

1. \*\*Components\*\* → \*\*Emphasis\*\* → \*\*Inversity\*\* → \*\*Lightness\*\* → \*\*Primitives\*\*
2. \*\*Components\*\* → \*\*Inversity\*\* → \*\*Lightness\*\* → \*\*Primitives\*\* (for non-interactive elements)

**Validation Rules:**

* Semantic layers should never directly reference primitives (except for lightness layer)
* Component layers should never reference lightness or primitive layers directly
* Each layer should only reference the layer immediately below it in the hierarchy

### Folder Structure vs. Token References

**Important:** Changing folder structures does not necessarily break token references. Token resolution depends on the token paths and names, not the folder organization.

**What Breaks References:**

* Changing token paths (e.g., `{ob.s.color.bg.default}` → `{ob.s.color.background.default}`)
* Renaming tokens (e.g., `primary` → `accent`)
* Removing tokens that are referenced elsewhere

**What Doesn't Break References:**

* Moving files between folders (as long as token paths remain the same)
* Reorganizing folder structure for better organization
* Renaming folders (token paths are independent of folder names)

**Theme Resolution ("Last Wins"):**

When multiple themes define the same token path, the last loaded theme takes precedence. This allows for theme overrides and layering without breaking references.

**Example:**

// Theme A defines: ob.s.color.bg.primary = "blue"

// Theme B defines: ob.s.color.bg.primary = "red"

// Result: Theme B wins, token resolves to "red"

This behavior enables flexible theme switching and customization without requiring reference updates.