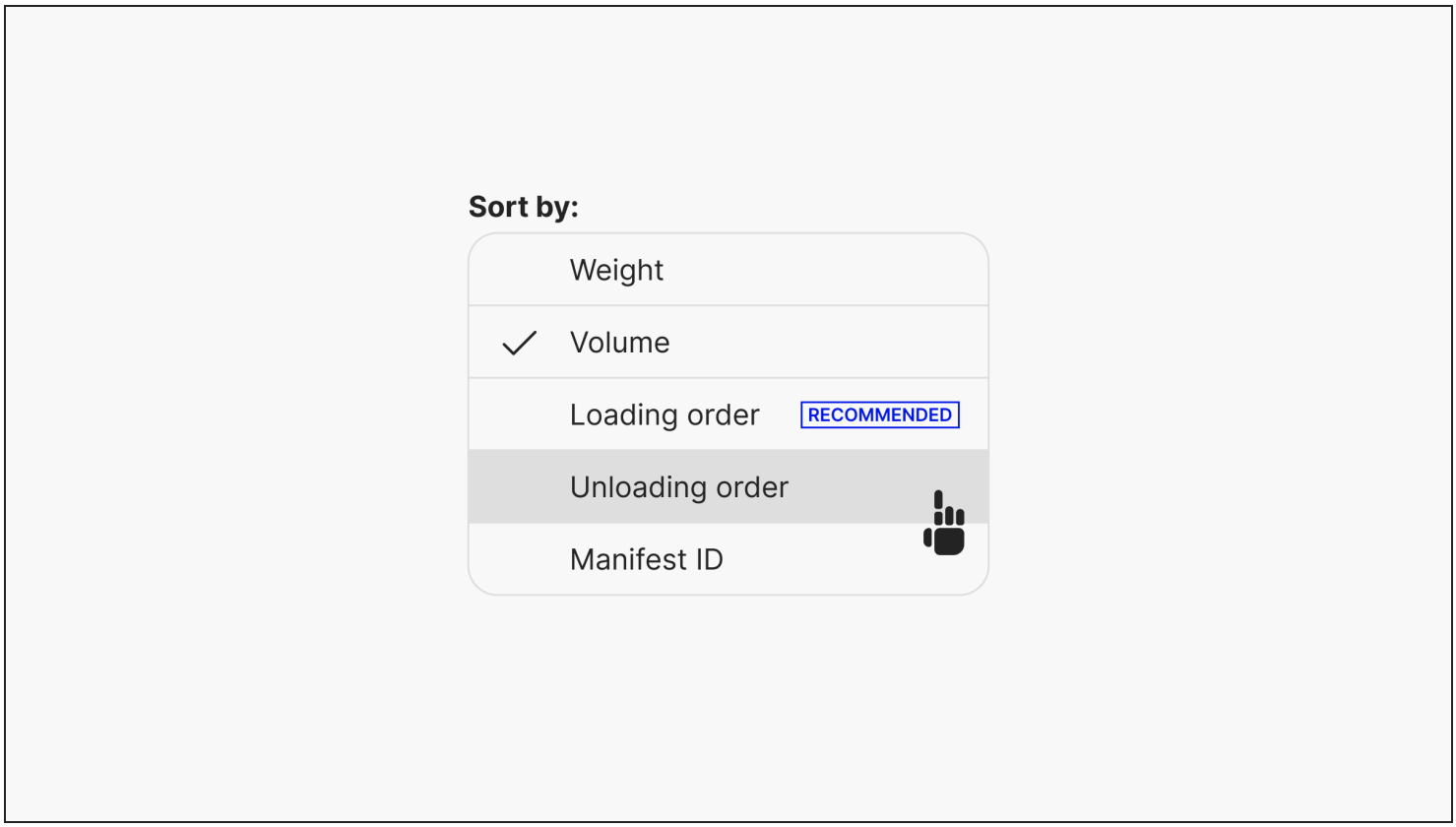
Taxonomy for Component States

Overview

States are visual feedback for a component's temporary condition. Users rely on their cues to understand and manipulate the UI. Well-designed states support accessibility, improve usability, and contribute to brand expression and delight.



Select menu demonstrating which option is currently selected and which option is currently hovered.

This document is a quick reference to inventory and label common states, in order to:

- Discuss within a design team and its technical partners
- Scope the work
- Design in a consistent, complete and effective manner
- Structure Figma assets

Methodology

Scope

States are a broad notion. This document covers user-triggered component's states.

This excludes application states, that are driven by data status, business logic, system services or back-end calls. It also excludes states that could be displayed on components, but aren't in response to direct user manipulation.

For example, the "Unread" indicator of a "Messages" tab isn't in scope. While the "Unread" status does exist, it characterizes a data attribute (read vs unread) rather than a component feedback (idle vs pressed down).

Component states are also different from components properties such as size, emphasis, theme etc. Those properties are set at design time and don't change at runtime.

User-triggered states are transient: they cycle in quick succession. Thus a good rule of thumb to tell them apart from other attributes is whether they reset on page load. If yes, they are likely components' states, as covered in this document.

Conventions

States are organized into "state categories". Each state name is unique.

Sources

– **All the user-facing states**
– **W3C**
– Major vendors component libraries and guidelines

Taxonomy

Pointer State

`Idle`, `Hovered`, `Pressed Down`, `Disabled`, `Loading`

Colloquially called "state". although only a subset of it.

"Pressed down" is preferred to "Active", as per WCAG.

Selection State

`Unselected`, `Selected`, `Indeterminate`

Focus State

`Unfocused`, `Focused`

Input State

`Empty`, `In progress`, `Filled`

Note that “Empty” means “empty of user input”, not “empty of any content”. There may still be a placeholder, as a component property.

Validation State

`No validation`, `Instructions`, `Valid`, `Invalid`

Note that “Instructions” means “explanations in reaction to user input”, not persistent help text. There may still be persistent help text, independently from instructions, as a component property.

Others

The states listed above are the most common. Further states should be considered on a case-by-case basis, such as “visited” for links or “dragged” for draggable elements.

How-To

Here’s how to use the provided taxonomy to support design activities.

Scope

For each component, assess which states from the taxonomy do apply. Not all components have all states: for example, a button has no notion of being “selected” or not. A radio button can be selected, but can’t have an “indeterminate” state.

State categories are multiplicative rather than additive. A tab control gets “Pointer Feedback” (4 applicable states) and “Selection feedback” (2 applicable states). That’s 4 × 2 = 8 states, not 4 + 2 = 6 states.

Note that that this step isn’t about “adding” states. Rather, it is about making sure existing states are accounted for, so that design decisions can be made with proper context.

Design

Just because a state exists doesn’t mean it needs a unique design. Some states can be very similar or downright identical in appearance.

States don’t have to be encoded with color only: icons, text, position shift etc. can also be very effective.

Finally, consider the whole component when styling states, like text labels associated to atomic controls.

Figma

Name properties or variants with the state categories and states from the taxonomy. This improves consistency for the builder and for the user, in turn improving maintainability and understandability. It is ok to omit the word “state” from category names.

Expose states as variants rather than booleans.