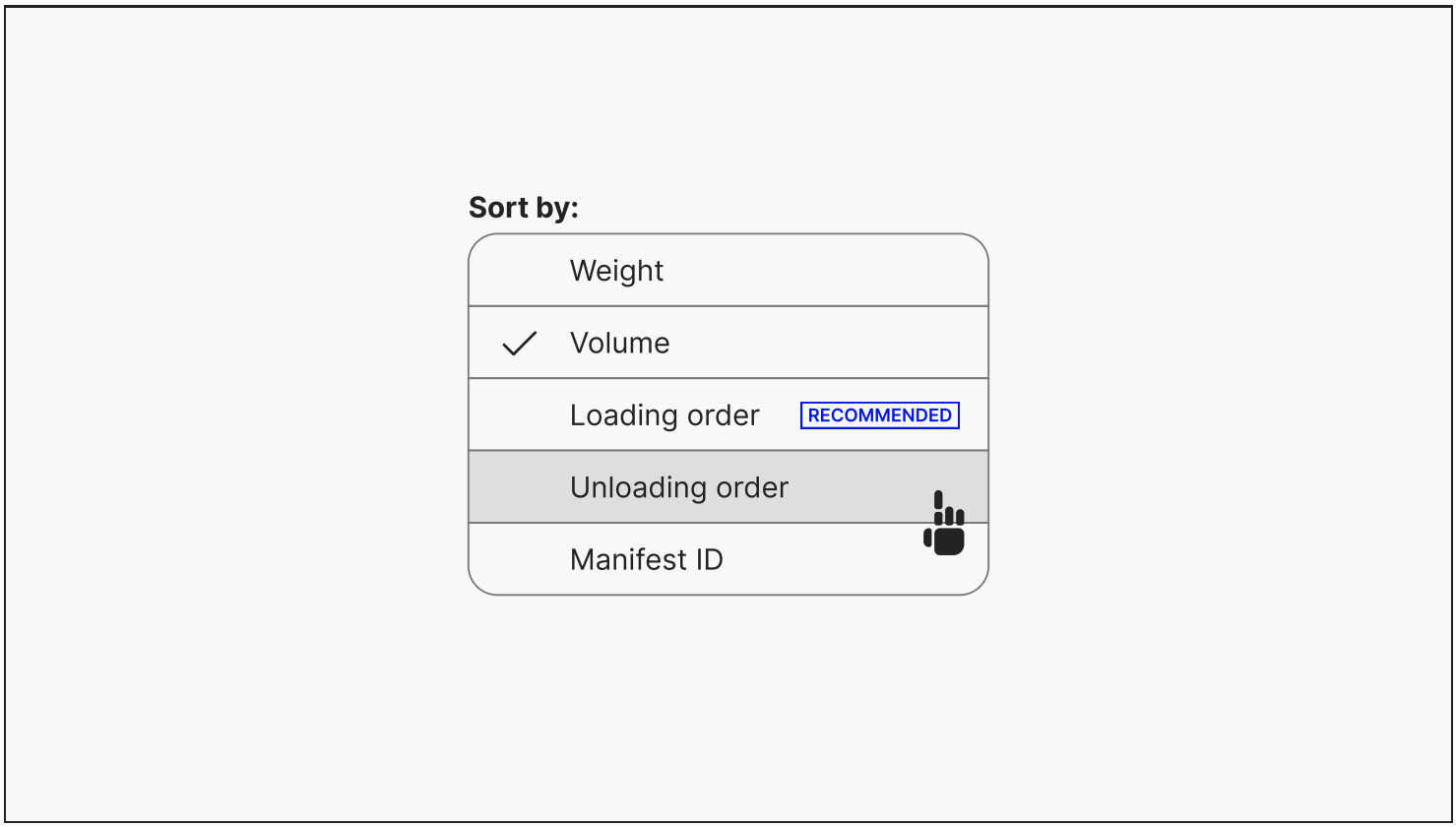
Taxonomy for Component States

Skip to Recap

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 options that are `idle, unselected`, `idle, selected` and `hovered, unselected`.

This document inventories and labels common states, in order to:

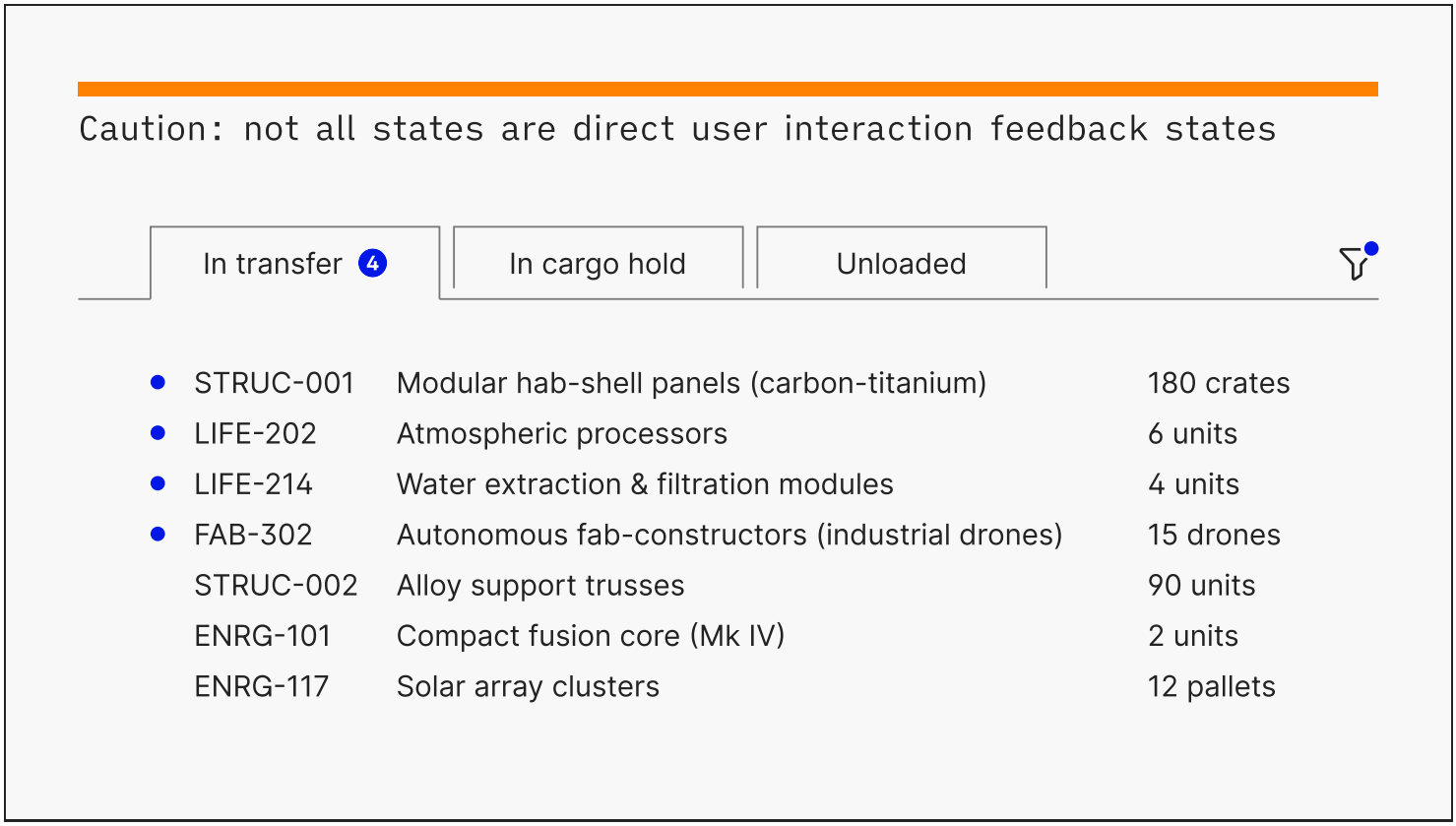
- Discuss within a design team and its technical partners
- Scope the work
- Design in a consistent, complete and effective manner
- Structure Figma assets

Methodology

Scope

States are a broad notion. This document covers direct user interaction feedback component's states.

This excludes application states, that are driven by data status, business logic, system services or back-end calls: incomplete, running or error progress statuses, new/undread, locked/unlocked, unsaved, etc. Those application states may sometimes be displayed on components, but they aren't a response to direct user manipulation.



For example, the "4 new" indicator of the "In transfer" tab isn't in scope. While the "new" status does exist, it characterizes a data attribute (`Read` vs `Unread`) rather than a component feedback (`Idle` vs `Pressed down`). Likewise, the "active" indicator on the filter communicates an application state, that only happens to be depicted on this component. Both indicators are changed not by interacting directly with the component, but by performing actions within the panel associated to it.

Furthermore, 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

In this taxonomy:

- States are organized into state categories
- Each state name is unique

– Within a category, states are sorted by frequency of use; the first one being the default value

Taxonomy

Pointer States

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

A pointer is a hardware-agnostic representation of input devices that can target a specific coordinate on a screen. This category covers the immediate feedback from mouse, touch or stylus interaction. It's colloquially called "state", although it's only a subset of them.

Selection States

`Unselected`, `Selected`, `Indeterminate`

Focus States

`Unfocused`, `Focused`

Input States

`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 States

`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.

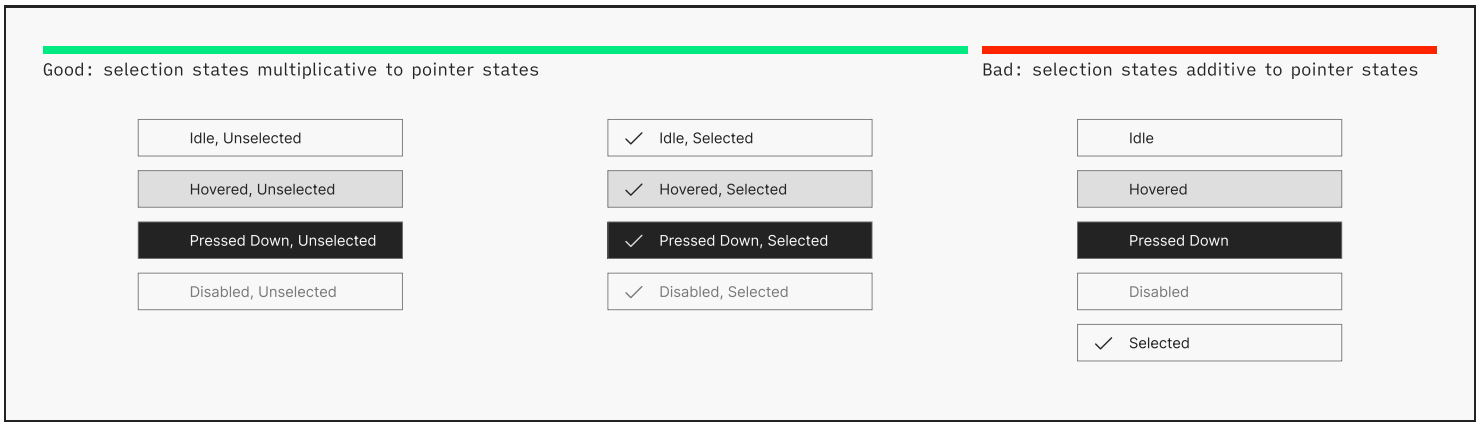
How-To

Using the provided taxonomy to support design activities.

Planning

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 select menu item gets “Pointer Feedback” (4 applicable states) and “Selection feedback” (2 applicable states). That’s 4 × 2 = 8 states, not 4 + 2 = 6 states.

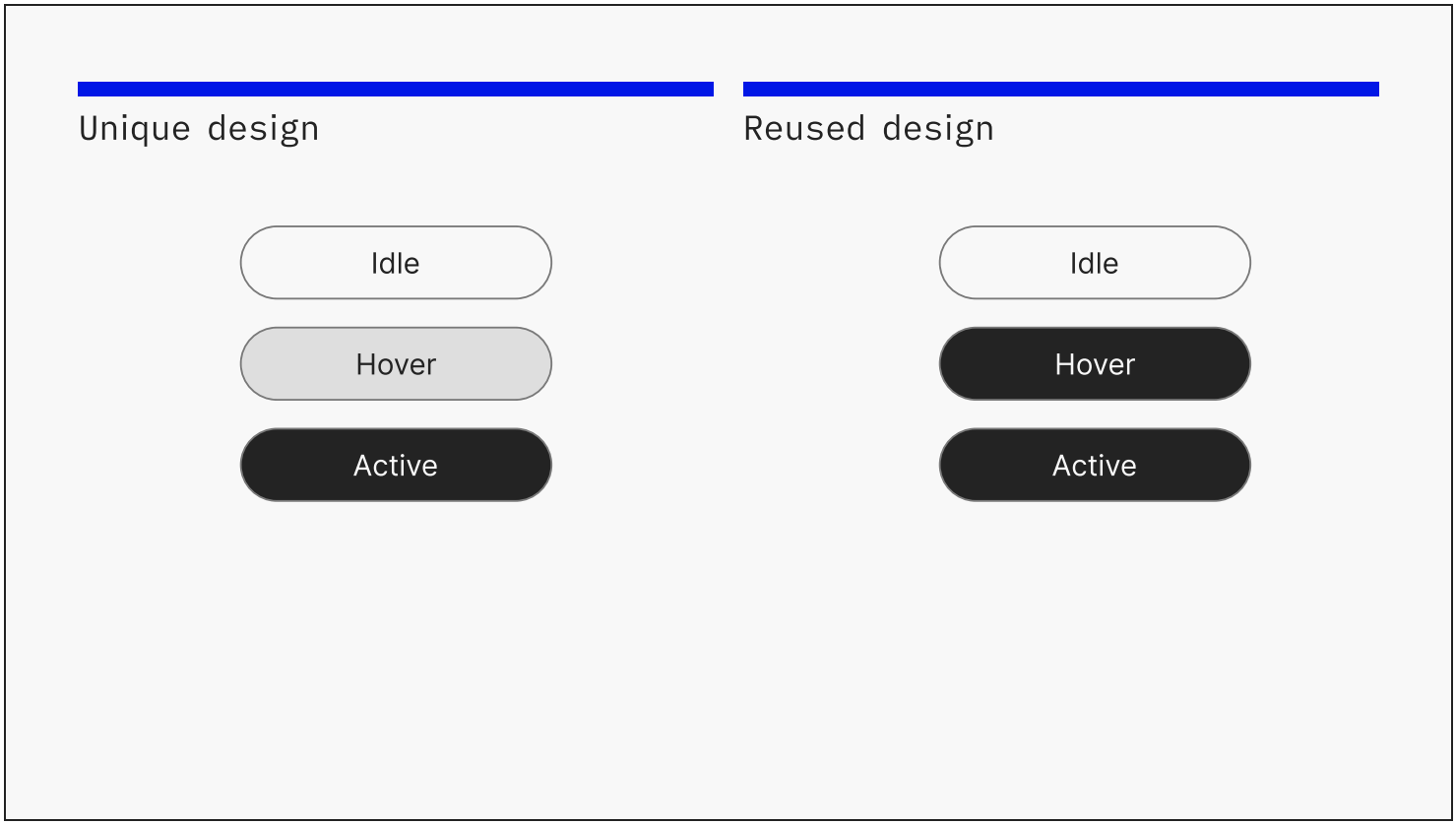


Comparison of multiplicative and additive state categories for select menu items.

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 is applicable doesn’t mean it needs a unique design. Some states can be very similar or downright identical to one another in appearance.



Conversely, designs don’t have to be a linear combination of prior choices. Some states can have a unique design, especially to better support specific micro-interactions.

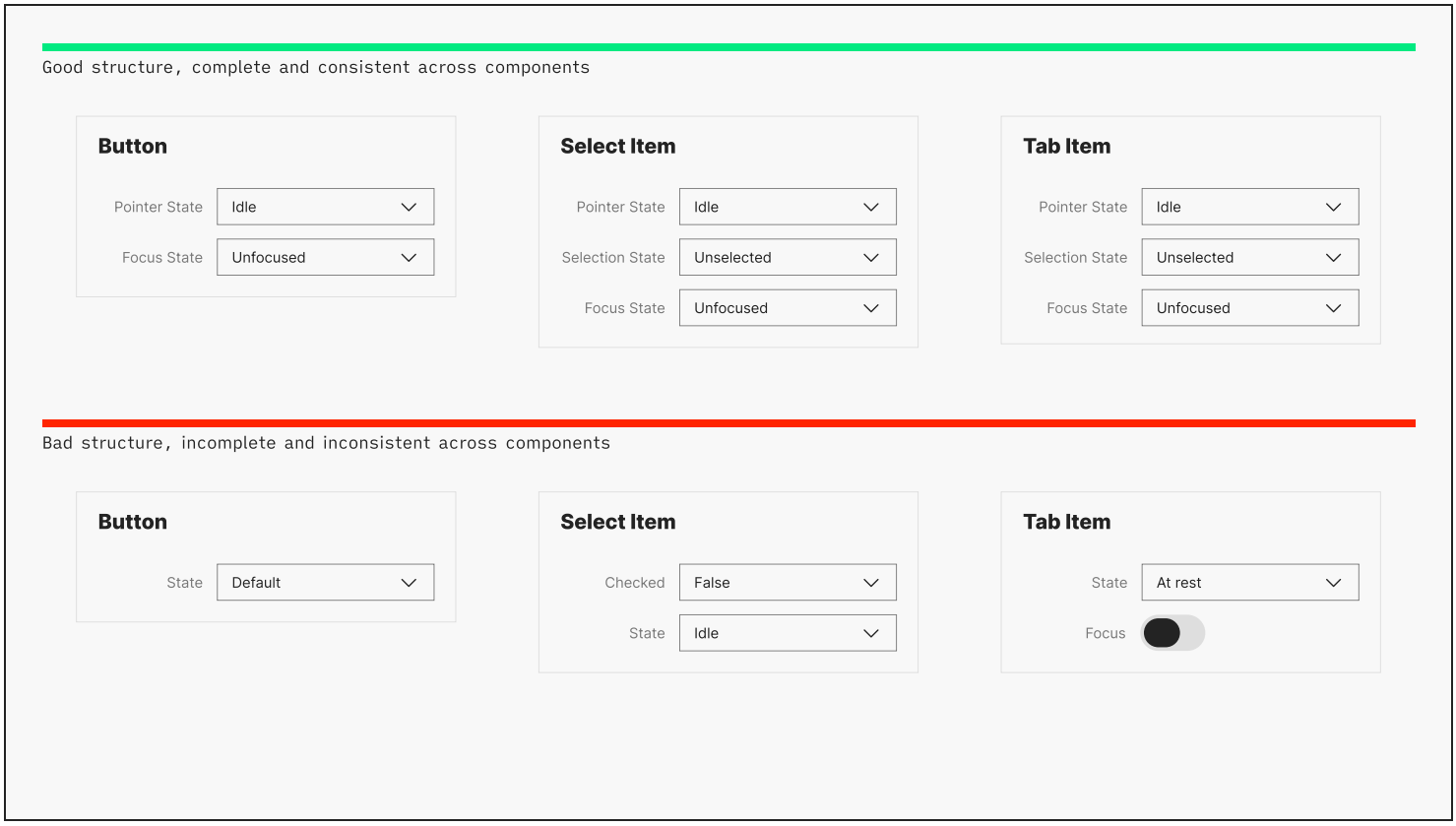
The combination of icon, color, size and shadow (left) conveys the `Selected` state much better than if only an icon had been used (right). Motion could have been used as well.

Finally, be sure to utilize all the possibilities of the target platform\: pointers, haptics, audio cues, etc.

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’s ok to omit the word “state” from category names.

Expose states as variants rather than booleans.



Comparison of well-structured and badly-structured component properties in Figma.

Sources

- **All the user-facing states**
- **cursor - CSS | MDN**
- **Pointer Events** from W3C
- **Web Content Accessibility Guidelines (WCAG) 2.1** from W3C
- Vendors component libraries and guidelines

Recap

Taxonomy for component states:

Pointer States: `Idle`, `Hovered`, `Pressed Down`, `Disabled`, `Loading`

Selection States: `Unselected`, `Selected`, `Indeterminate`

Focus States: `Unfocused`, `Focused`

Input States: `Empty`, `In progress`, `Filled`

Validation States: `No validation`, `Instructions`, `Valid`, `Invalid`

Others: `Visited`, `Dragged`, etc on a case-by-case basis