

Designing Interfaces with Components and States

- Shift from static screens to reusable, state-aware component architecture
- States replace page-by-page design with systematic, predictable behavior
- Improves designer-developer handoff, UI consistency, and user understanding
- Roadmap: discover, define, compose, document, and scale stateful components



1 DISCOVER

Understand needs, users, and existing patterns.



2 DEFINE

Define component structure, variants, and states.



3 COMPOSE

Build components that are flexible and scalable.



4 DOCUMENT

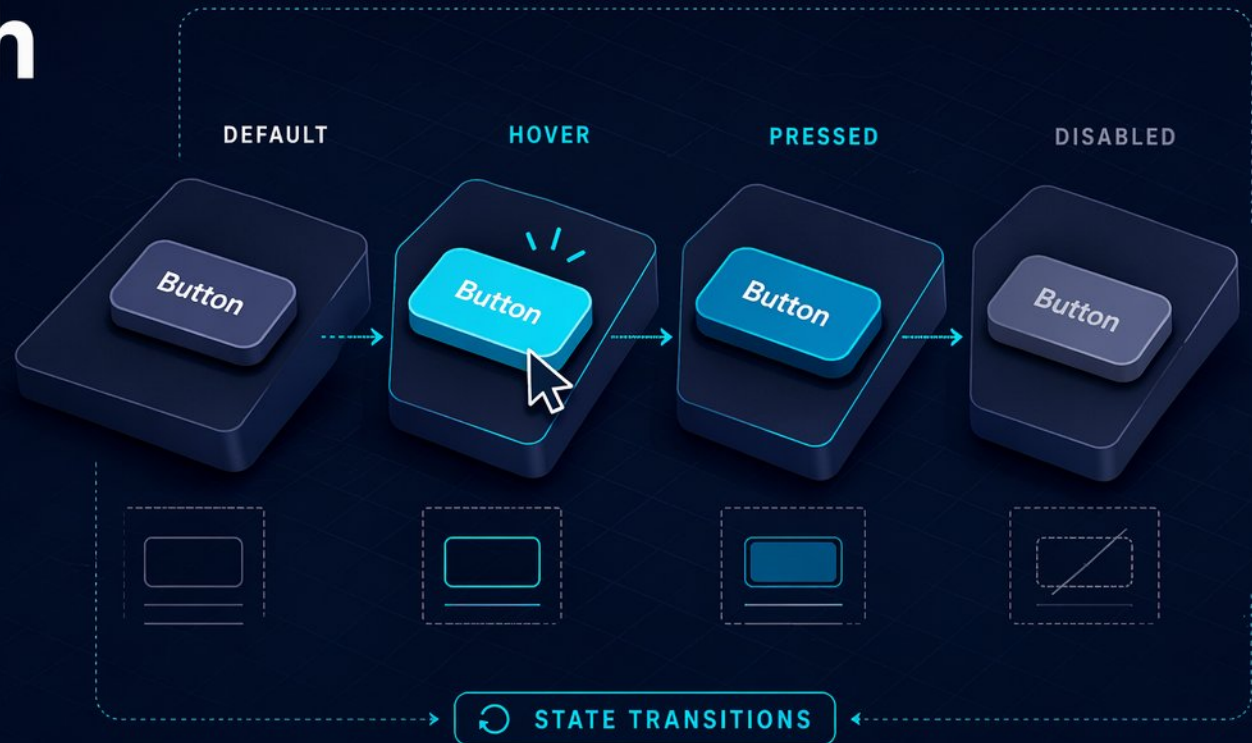
Create clear guidelines, examples, and dos & don'ts.



5 SCALE

Evolve the system and drive adoption.

ONE COMPONENT. MANY STATES.



The Mental Model: Components as State Machines



- A component is a container with a finite set of visual and behavioral states



- States are mutually exclusive: default, hover, active, disabled, loading, empty, error, success



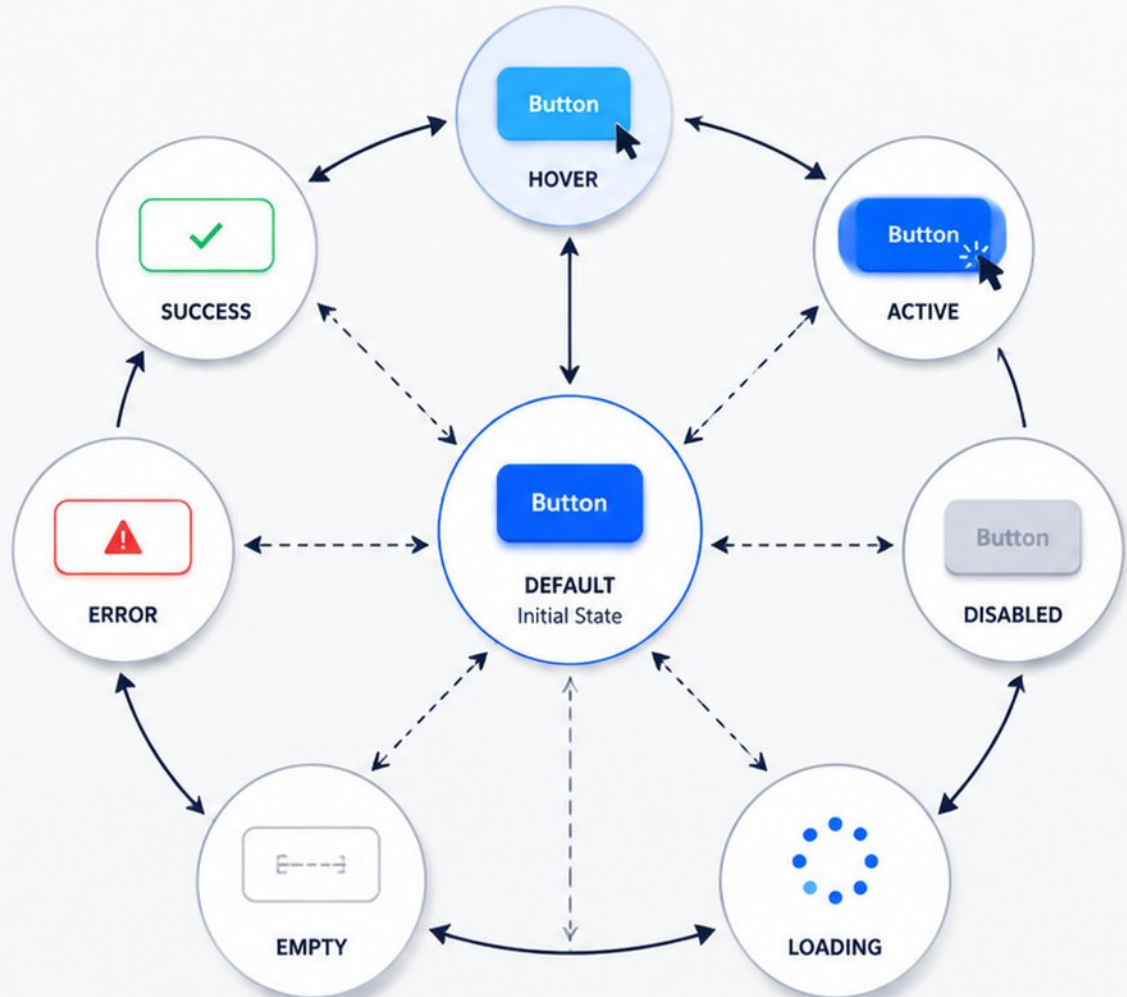
- State thinking replaces static screen-by-screen design with dynamic system design



- Design components directly parallel front-end component patterns (React, Vue, web components)



BUTTON COMPONENT STATE MACHINE



The 5 Core Interactive States You Can't Skip

1

Default



- Default: the resting, baseline appearance of the component

2

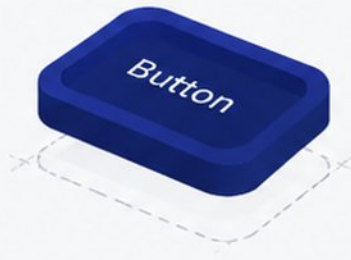
Hover



- Hover: pointer affordance on desktop; never rely on it for touch

3

Active/Pressed



- Active/Pressed: momentary feedback on click or key activation

4

Focus



- Focus: high-contrast indicator critical for keyboard navigation

5

Disabled



- Disabled: clearly non-interactive with perceivable, reduced affordance

Beyond the Basics: Data-Driven and Feedback States



- **Loading:** spinners, skeleton screens, and progress bars; announce with aria-busy



- **Empty:** explain why it's empty and provide a clear next action; never just blank space



- **Error:** highly visible, clear explanation, and a path to recovery; never color alone

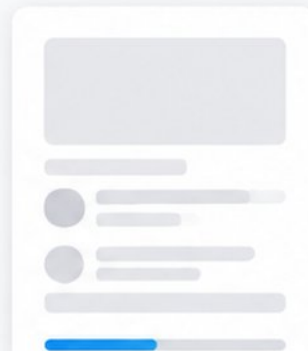


- **Success:** transient confirmation that an action completed



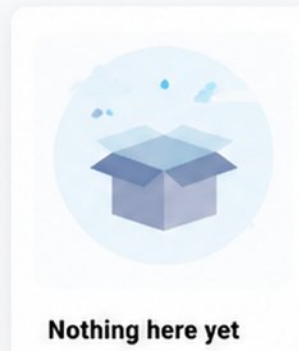
- **Partial/Incomplete:** zero-data and partial-fill treatments for content components

COMMON FEEDBACK STATES IN A CARD COMPONENT



Loading

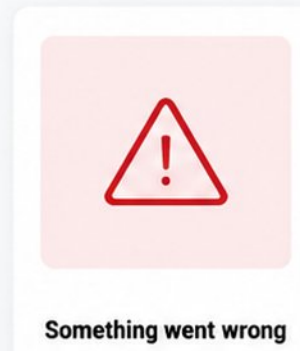
We're getting things ready for you...



Nothing here yet

This is empty because you haven't added any items.

[Add your first item](#)

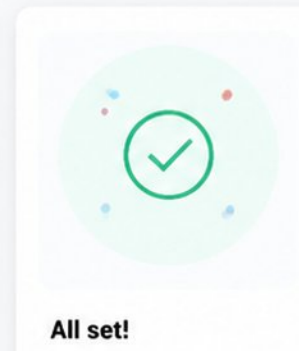


Something went wrong

We couldn't load this content. Please try again.

[Try again](#)

[Contact support](#)



All set!

Your changes have been saved.

STATE TRANSITIONS



Loading

System is working or fetching data



Empty

No data to show yet



Error

Something failed to load



Partial/Incomplete

Some data available or partially filled



Success

Action completed successfully



The State Matrix:

Crossing States with Variants



- Interaction states multiplied by approved variants creates the full matrix



- Every cell must be deliberately designed, not discovered in production



- Semantic tokens drive consistent state styling across all variants

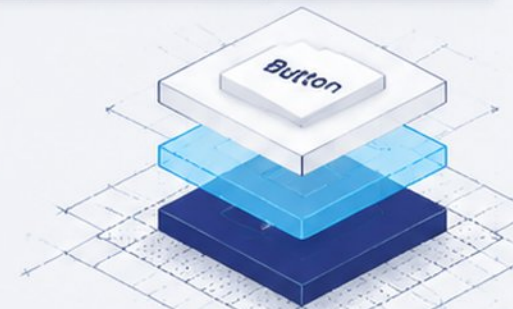


- If the matrix balloons, decompose the component or use composition

		INTERACTION STATES				
		 Default	 Hover	 Active	 Selected	 Disabled
SIZE VARIANTS	 Extra Large	Button	Button	Button	Button	Button
	 Large	Button	Button	Button	Button	Button
	 Medium	Button	Button	Button	Button	Button
	 Small	Button	Button	Button	Button	Button



Every cell is a deliberate design decision



Discovering Component Boundaries



- Identify components by single responsibility, data alignment, and reuse signals



- Atomic Design (atoms, molecules, organisms) is a mental model, not a linear process



- Delay extraction until you have real reuse to avoid premature abstraction



- Audit existing products with component inventories and state tables

KEY

 Component Boundary

 Composition Flow

BOUNDARIES

Dotted lines define component boundaries—encapsulation enables reuse, consistency, and change isolation.



ATOMS

Fundamental building blocks



MOLECULES

Groups of atoms forming simple components



ORGANISMS

Complex sections of UI



PAGE LAYOUT

Assembled organisms to form complete screens



Mapping States for a Single Component



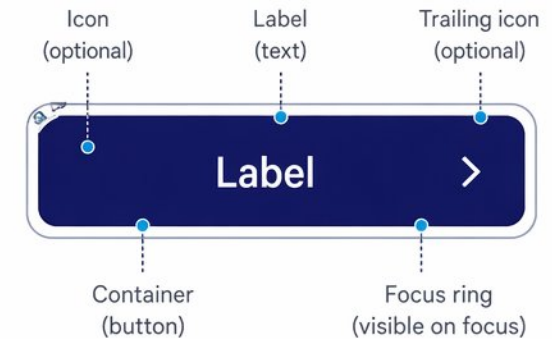
- Exhaustively list all states using a step-by-step audit: interactive, data, validation, and edge cases
- Use state inventory worksheets and trigger tables; columns: state, trigger, token, accessibility
- Apply canonical vocabulary: default, hover, focus, active, disabled, loading, empty, error, success
- Common missing states: focus-visible, loading, empty, error, and edge cases like long text or overflow



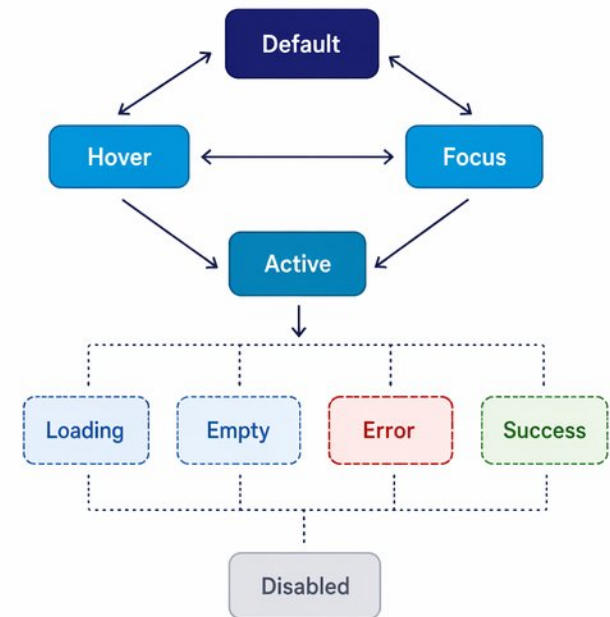
STATE INVENTORY WORKSHEET

STATE		⚡ TRIGGER	🏷️ TOKEN	♿ ACCESSIBILITY
Default		Page load / No interaction	btn/default	Meets contrast AA (normal text)
Hover		Pointer over	btn/hover	Meets contrast AA (normal text)
Focus		Tab / Keyboard focus	btn/focus	Visible focus indicator (2px)
Active		Click / Key down (Enter or Space)	btn/active	Meets contrast AA (normal text)
Disabled		Disabled state or unavailable	btn/disabled	Meets contrast AA (disabled)
Loading		Async action in progress	btn/loading	Announce status (aria-busy="true")
Empty		No data available	btn/empty	Informative text provided
Error		Action failed / Error state	btn/error	Error identified aria-invalid="true"
Success		Action succeeded / Completed	btn/success	Status message available

COMPONENT ANATOMY

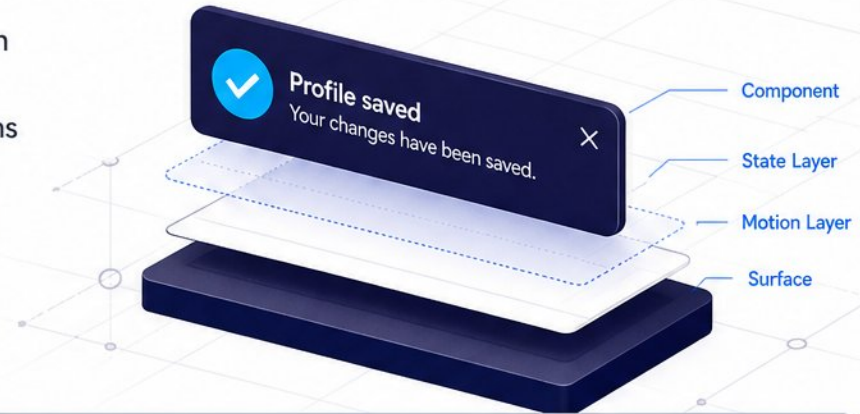


STATE TRANSITIONS



Transitions and Lifecycles

- State changes communicate through meaningful motion, not decoration
- Mount, update, and unmount phases each have distinct design implications
- Design for interrupted or failed transitions; never leave a component stuck
- Specify animation triggers, duration, and easing explicitly for handoff



TOAST NOTIFICATION LIFECYCLE



TRIGGERS

Events that start transitions



DURATION

Define time in milliseconds



EASING

Control acceleration curves



INTERRUPTION

Handle cancel and rollback

Composing Components into Interfaces



- Nest components via parent-child boundaries that share state through context, not prop drilling



- Prefer slots and children for composition over configuration to avoid prop explosion

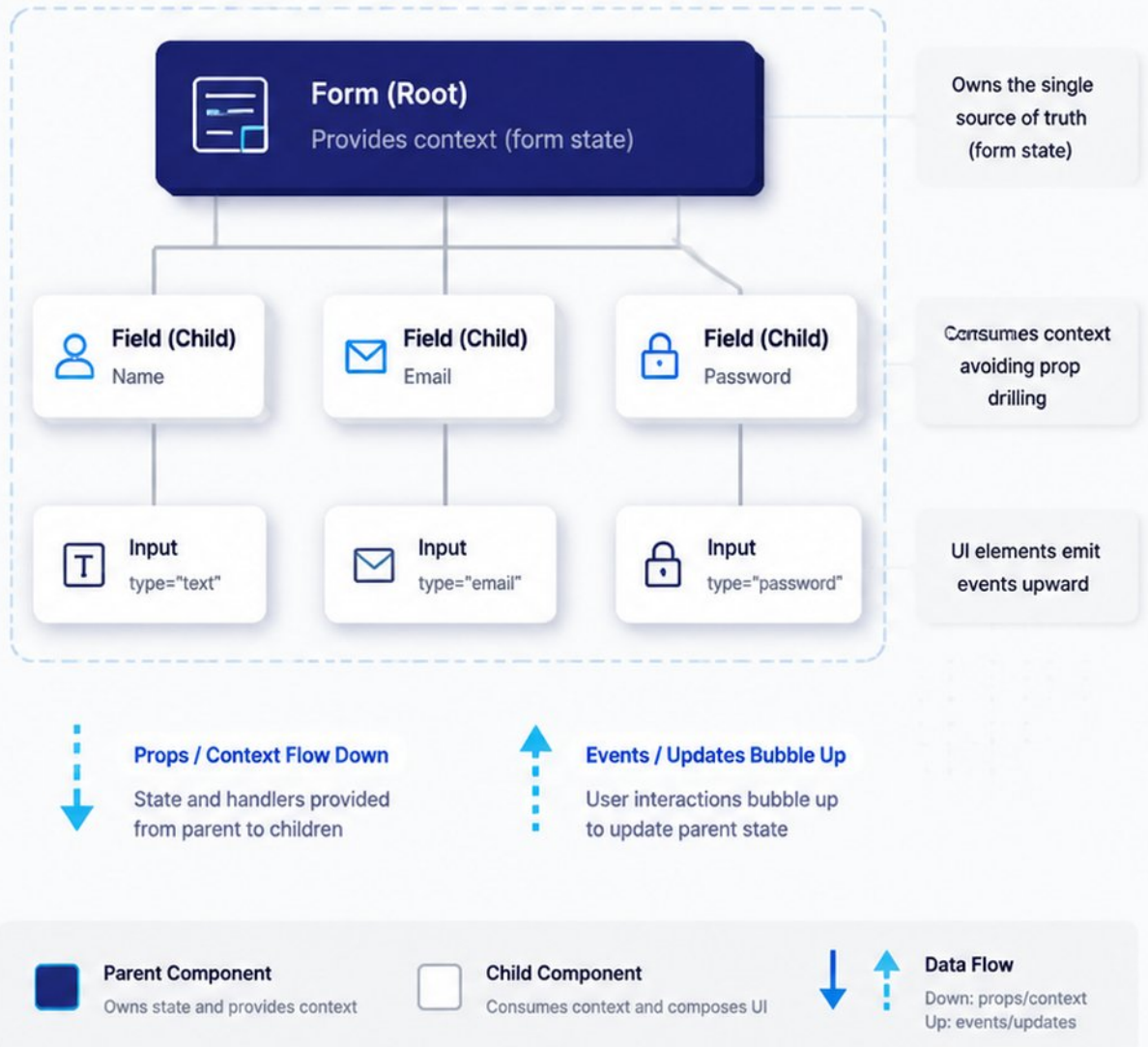


- Use compound component patterns (Root, Trigger, Content) for flexible, implicit state sharing



- Design specifications must prevent state duplication and prop bloat across nested components

Example: Form Component Hierarchy



Slot-Based Design and Composition Patterns



- Slot APIs give consumers content control; components handle layout and accessibility



- Props for simple variations; slots for unpredictable, open-ended customization



- Headless components separate logic from presentation for maximum flexibility



- Naming conventions: Root, Trigger, Content, Header, Footer, Title, Description

HEADER

TITLE

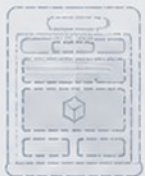
DESCRIPTION

CONTENT

FOOTER



STATE TRANSITIONS



EMPTY



PARTIALLY FILLED



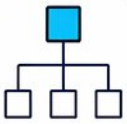
FILLED

Documenting Components and States for Handoff



01 What developers need:

- What developers need: all states, component mapping, interactions, responsive rules, and data specs



02 Structuring docs:

- Structuring docs: purpose, anatomy, props, states, variants, do's & don'ts, and accessibility



03 Tools & formats:

- Tools & formats: Figma annotations, Storybook, Zeroheight, and living documentation



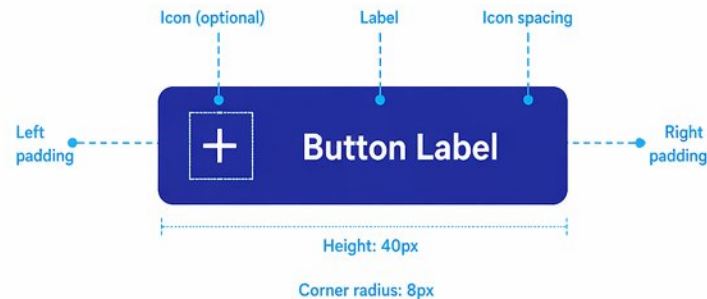
04 The handoff checklist:

- The handoff checklist: state completeness, focus visibility, edge case annotations, and contextual review

Component: Button

Primary action button used to trigger main workflows.

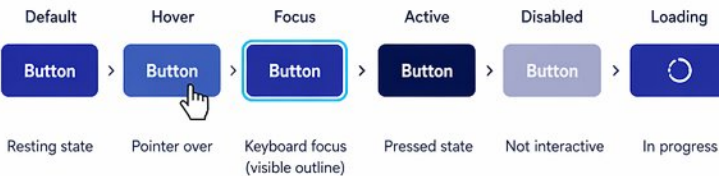
Anatomy



Props

Prop	Type	Default	Description
label	string	—	Text displayed in the button
variant	enum	primary	Style variant (primary, secondary)
size	enum	md	Size (sm, md, lg)
icon	node	—	Icon element (optional)
disabled	boolean	false	Disables the button

States



Accessibility

- Keyboard: Tab to focus, Enter/Space to activate
- Focus visible: 2px outline, 2px offset
- Screen reader: Announces label and state
- Contrast: Meets WCAG AA (4.5:1)

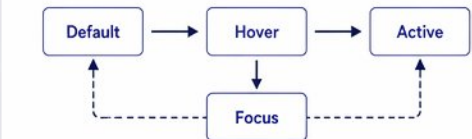
Variants



Do's & Don'ts

- Do**
 - Use for primary actions
 - Keep labels concise
 - Maintain sufficient contrast
- Don't**
 - Don't use for navigation
 - Don't use unclear labels
 - Don't disable without good reason

Interactions



Responsive Rules

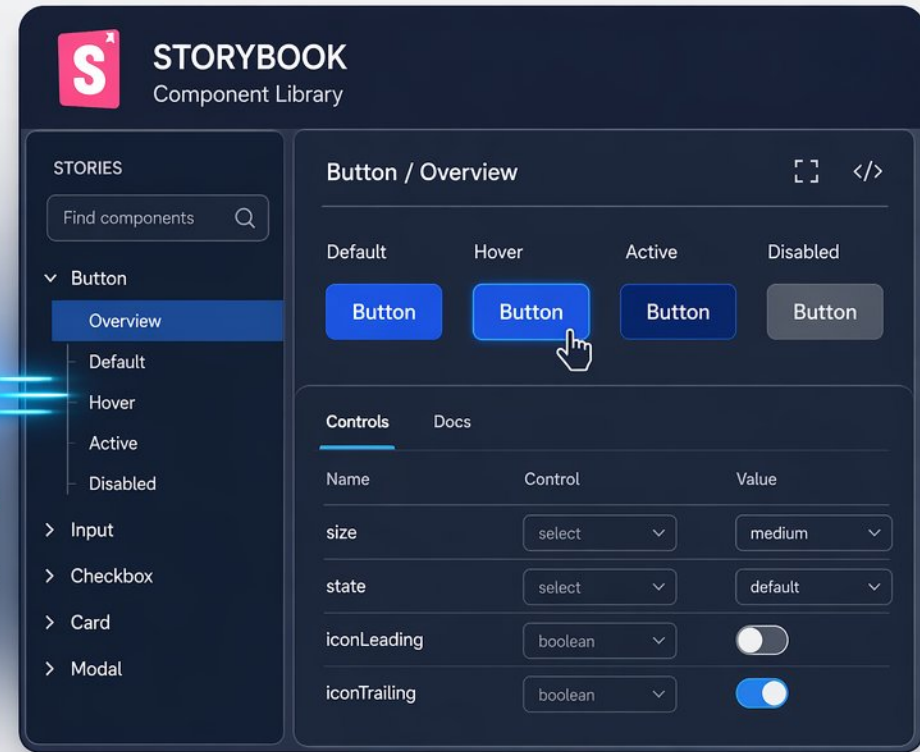
	Desktop ≥ 1024px	Size: md (40px height) Icon + label inline		Tablet 768px – 1023px	Size: sm (36px height) Icon + label inline		Mobile ≤ 767px	Full width button Icon + label inline
---	----------------------------	---	---	---------------------------------	---	---	--------------------------	--

Figma to Code: Aligning Design and Development



SINGLE SOURCE OF TRUTH

Component linked to Storybook



TOKEN TO PIXEL CASCADE



Use Figma Dev Mode, variables, and clear layer naming for handoff



Variants and component properties reveal state logic to developers



Annotate focus, precedence, overflow, and edge cases—not the obvious



Link Figma components to Storybook stories for a single source of truth

Testing Component Behavior



- **Design review:** verify all states, not just default, before handoff



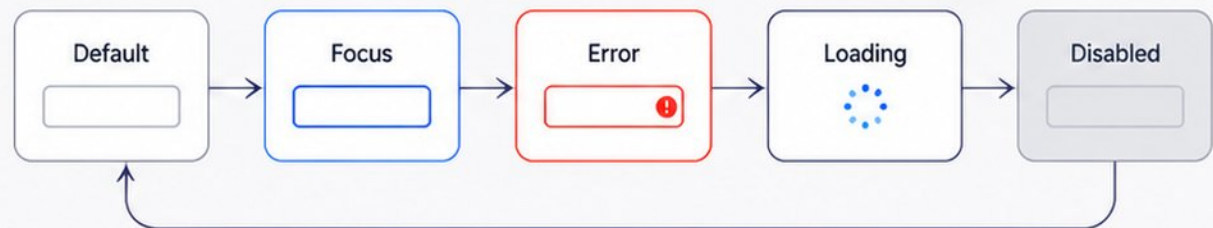
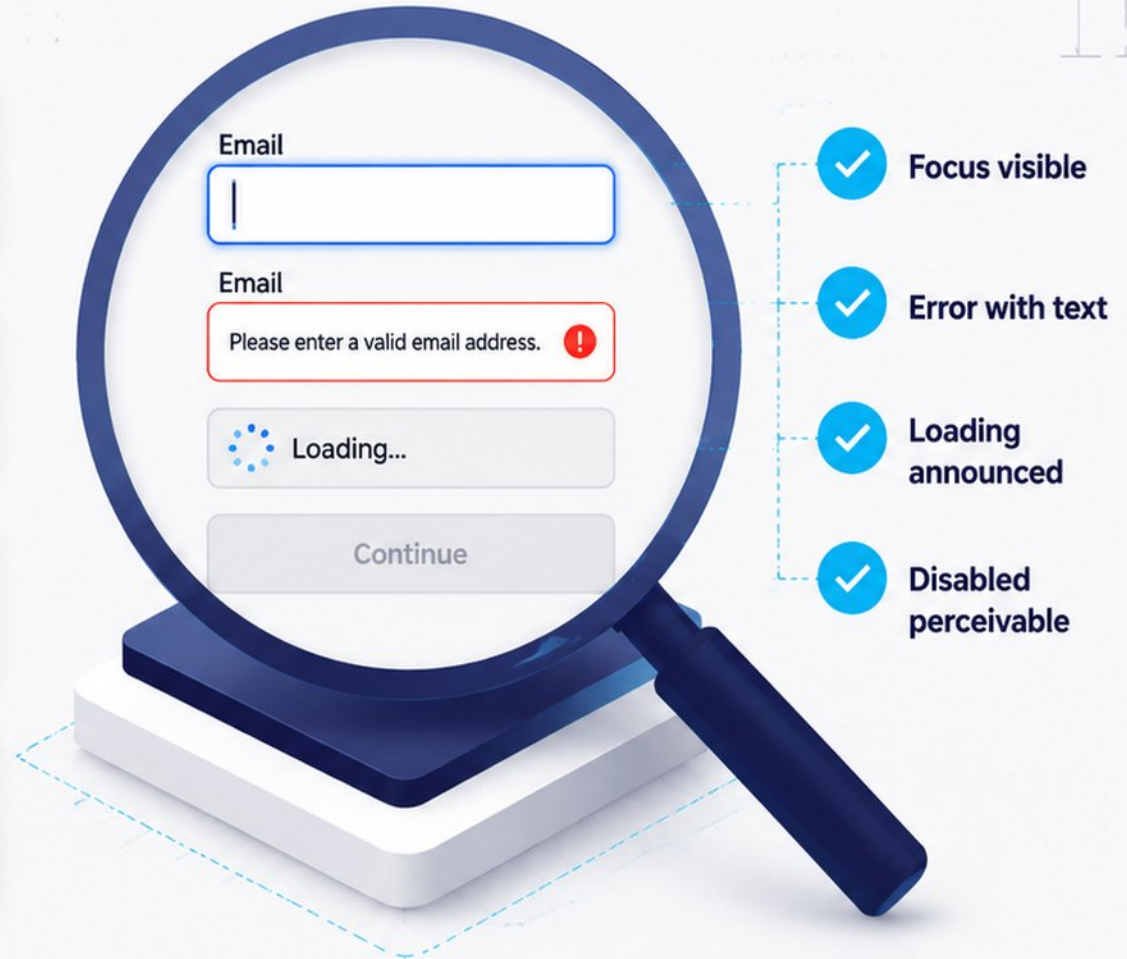
- **Acceptance criteria:** focus visible, error with text, loading announced, disabled perceivable



- **Visual regression testing:** catch unintended visual changes early

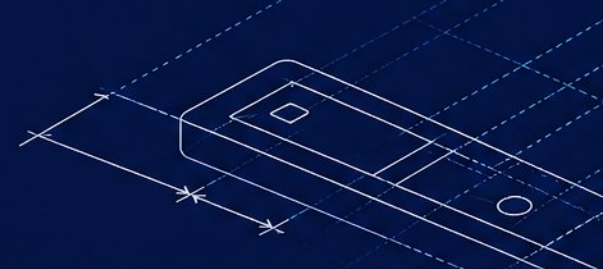


- **Prototype state transitions** to validate behavior before build





Accessibility Requirements for Every State



1



Focus: visible, high-contrast, distinct from hover; meets WCAG 2.4.7 and 2.4.11

2



Error: never color alone; include icon, text, and associate message with field

3



Disabled: perceivable by screen readers; explain why; consider read-only instead

4



Loading: announce with aria-busy or live regions; not just a visual spinner

5

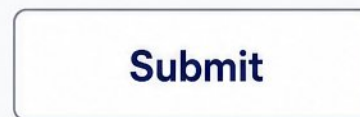


State testing: verify all states work without relying on color

SAME BUTTON, DIFFERENT STATES

DEFAULT

Normal state



Clear, standard appearance

FOCUS

Keyboard focus



High-contrast, distinct focus indicator

ERROR

Validation error



Icon + text error associated with the field

Please enter a valid email address.

LOADING

aria-busy="true"



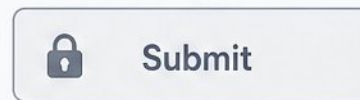
Announced via aria-busy or live regions



Submitting, please wait.
This may take a few seconds.

DISABLED

Not interactive



Perceivable to assistive tech; explain why



Submit unavailable. Form is read-only because the review period has closed.

Scaling to Design Systems



- Hybrid governance: core team owns foundations, product champions bring real use cases



- Design tokens as shared language: primitive → semantic → component tiers



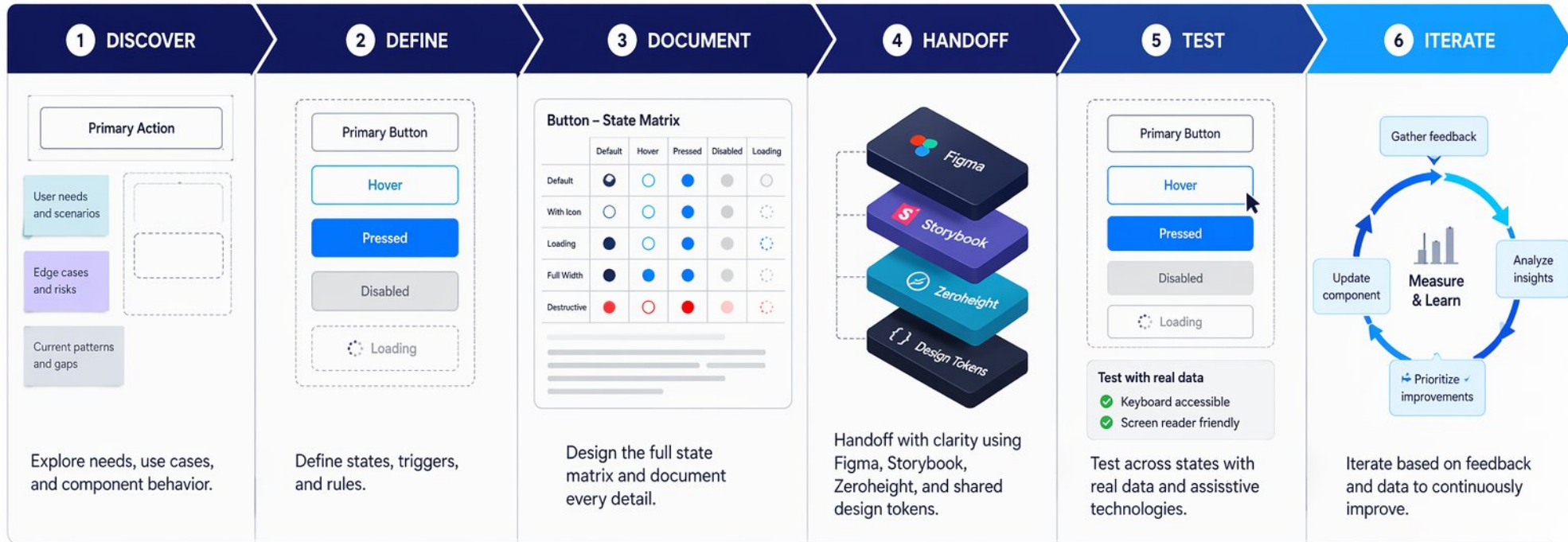
- Contribution path: propose, review, build, document, release, measure



- Track adoption: component usage, token health, detachment rates, accessibility defects



From Concept to Implementation: A Practical Workflow



Discover, define, document, handoff, test, and iterate on stateful components



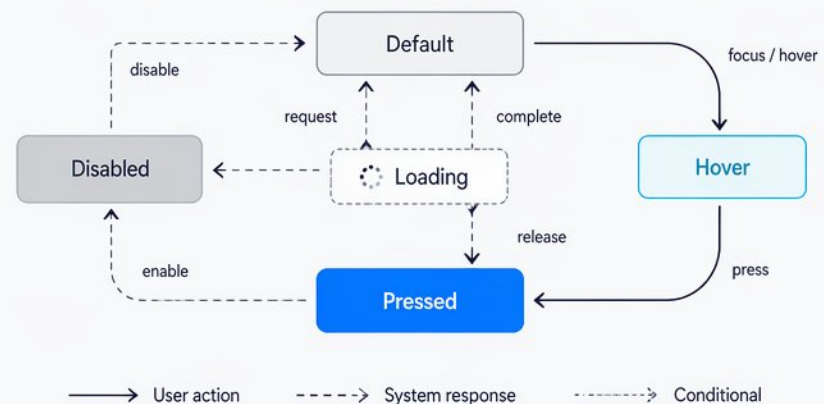
Integrate state design into existing sprints without slowing delivery



Align teams with Figma, Storybook, Zeroheight, and shared design tokens



Start with states, design the full matrix, document everything, test with real data



PersonWise.

PersonWise • AI digital-human course platform



Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/4034f1ec/public