

UI Design Framework:

Components and Application

- A governed rulebook, not just a component folder
- Formal principles, component libraries, and governance
- UI kits: parts; design systems: catalog + instructions
- Frameworks add principles, processes, and core rules
- For product designers, UI designers, and front-end teams



Why Systematic UI Pays Off



- 25-50% faster design and development; 47% faster time-to-market



- Consistency builds user trust across all product surfaces



- Less design debt, faster onboarding for new team members



- Accessibility fixes propagate everywhere: fix once, benefit everywhere



Anatomy of a UI Component



- Atomic hierarchy: atoms, molecules, and organisms — each layer matters



- A closed state set: default, hover, focus, active, disabled, loading, error

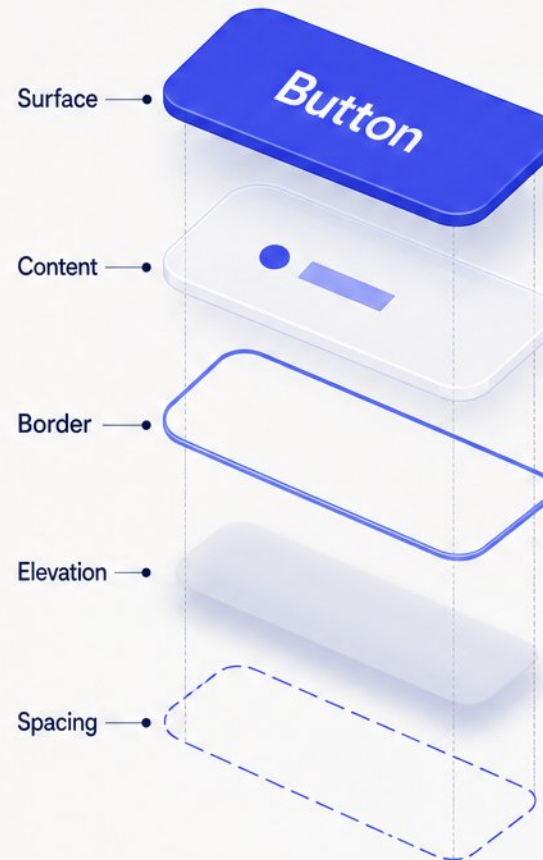


- Excessive variants and props create a tax, not a system



- Naming and documentation are part of the component contract

COMPONENT ANATOMY



STATE LAYERS

Default Button

Hover Button

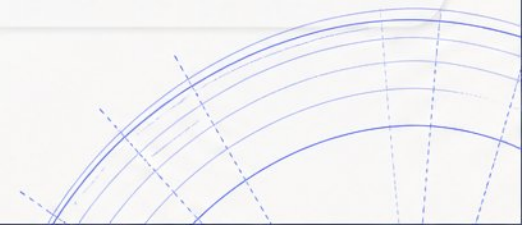
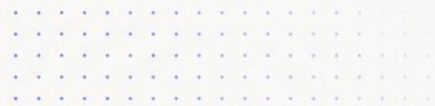
Focus Button

Active Button

Disabled Button

Loading Button

Error Button



Core Components Every Team Needs



- **Foundation primitives:** color, typography, spacing, elevation



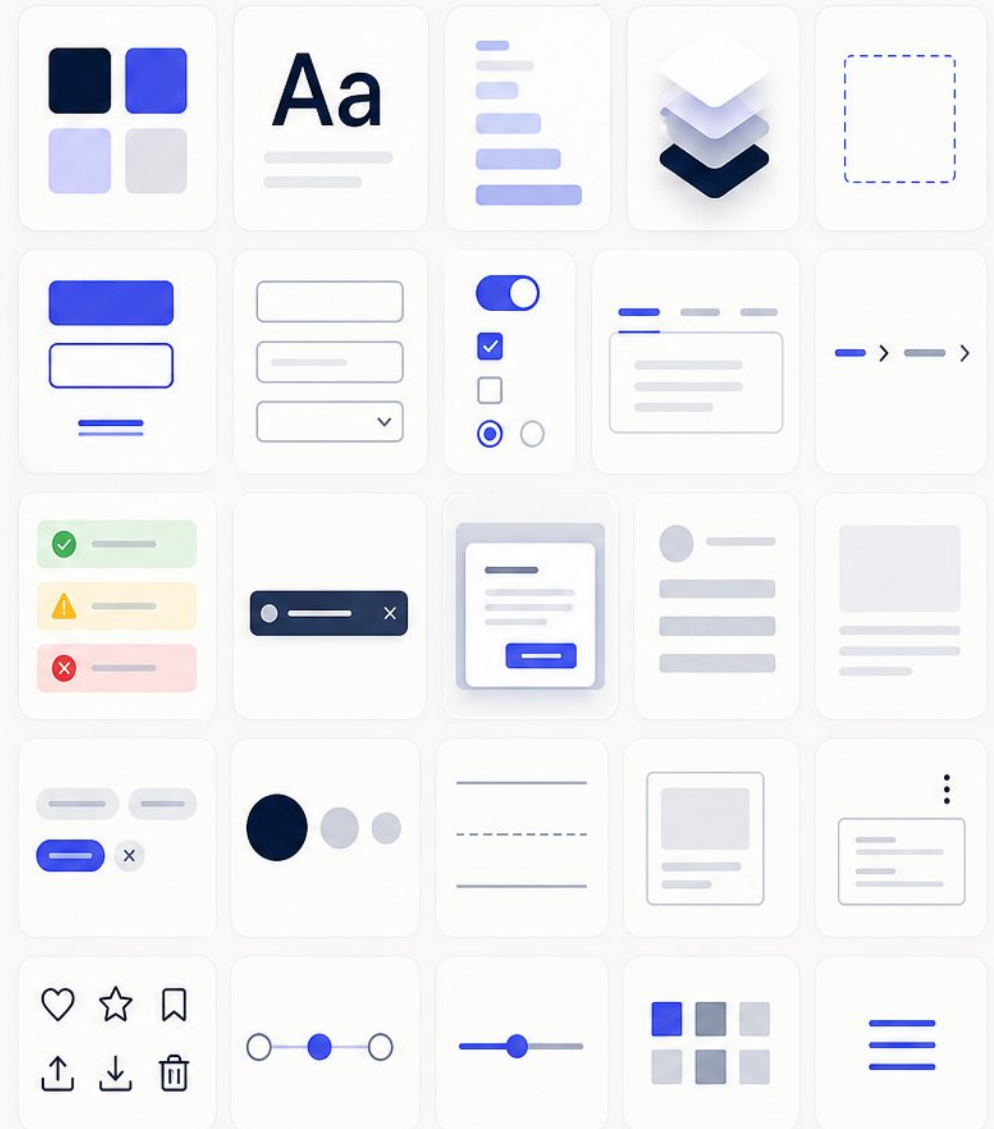
- **Input controls and navigation:** buttons, forms, tabs, breadcrumbs



- **Feedback and states:** alerts, toasts, modals, skeletons

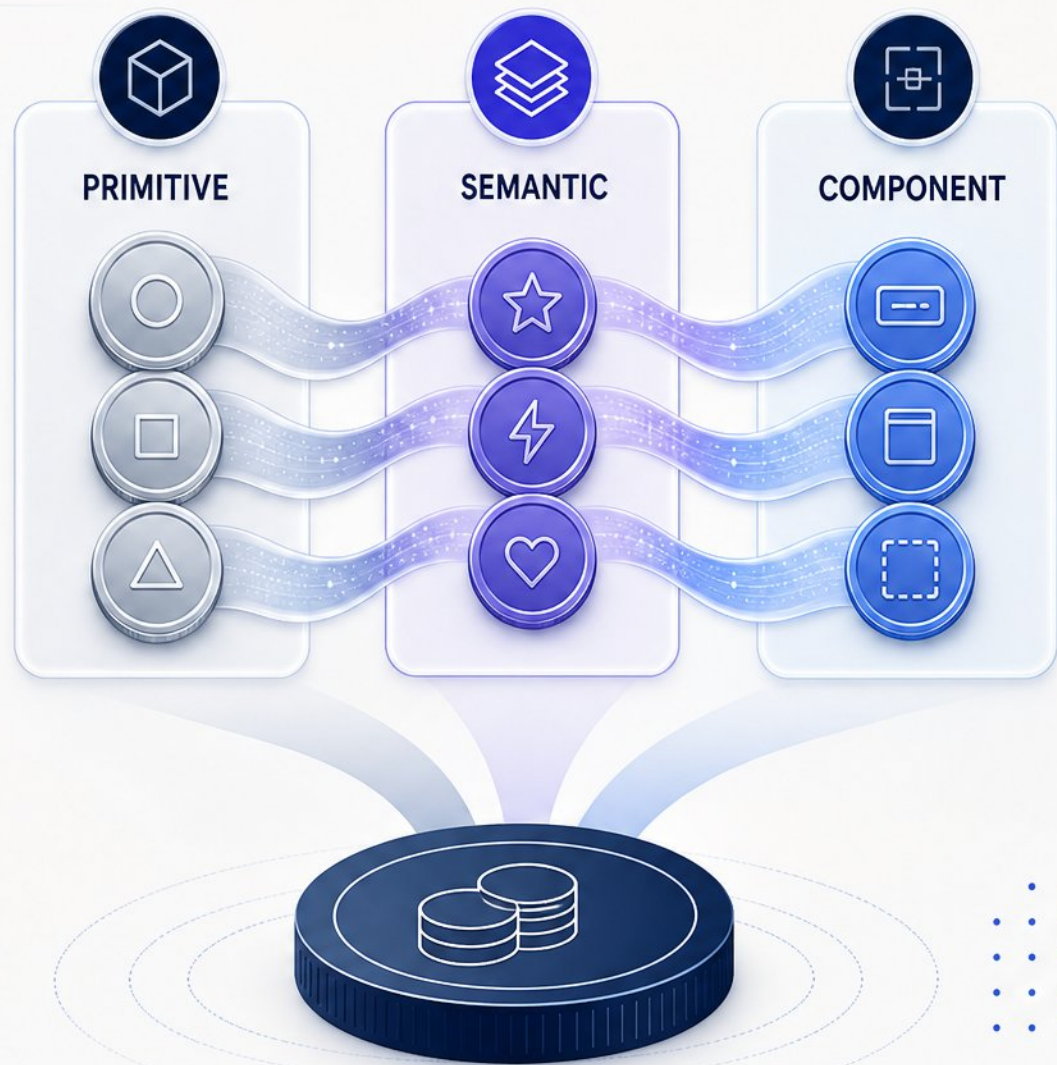


- **Start small:** 15-30 well-built components cover most of the UI

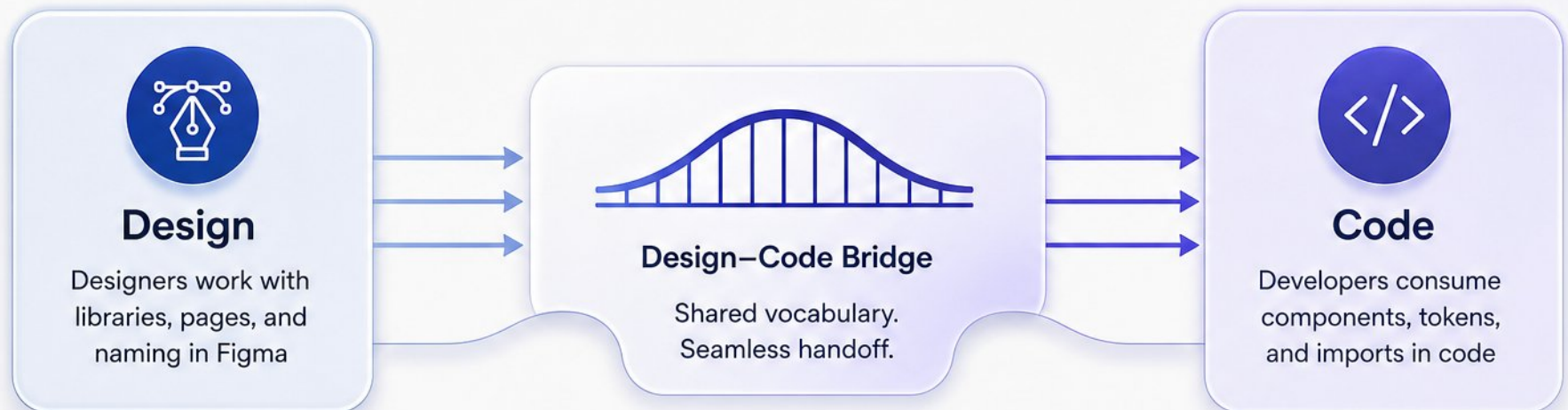


Design Tokens: The Shared Source of Truth

- Named design decisions, not raw values
- Three tiers: primitive, semantic, component — semantic matters most
- Semantic layer enables theming, dark mode, and multi-brand support
- Standard tools: W3C DTCG format, Style Dictionary, Tokens Studio
- Purpose-based naming (color-action-primary), not appearance-based



Using the Framework in Real Workflows



Designers work with libraries, pages, and naming in Figma



Developers consume components, tokens, and imports in code



Dev Mode and Code Connect link designs to production components



Design tokens sync as the shared source of truth



The real handoff gap is shared vocabulary, not tooling

Accessibility and Responsiveness in Components



- WCAG 2.2 AA baseline for every interactive component



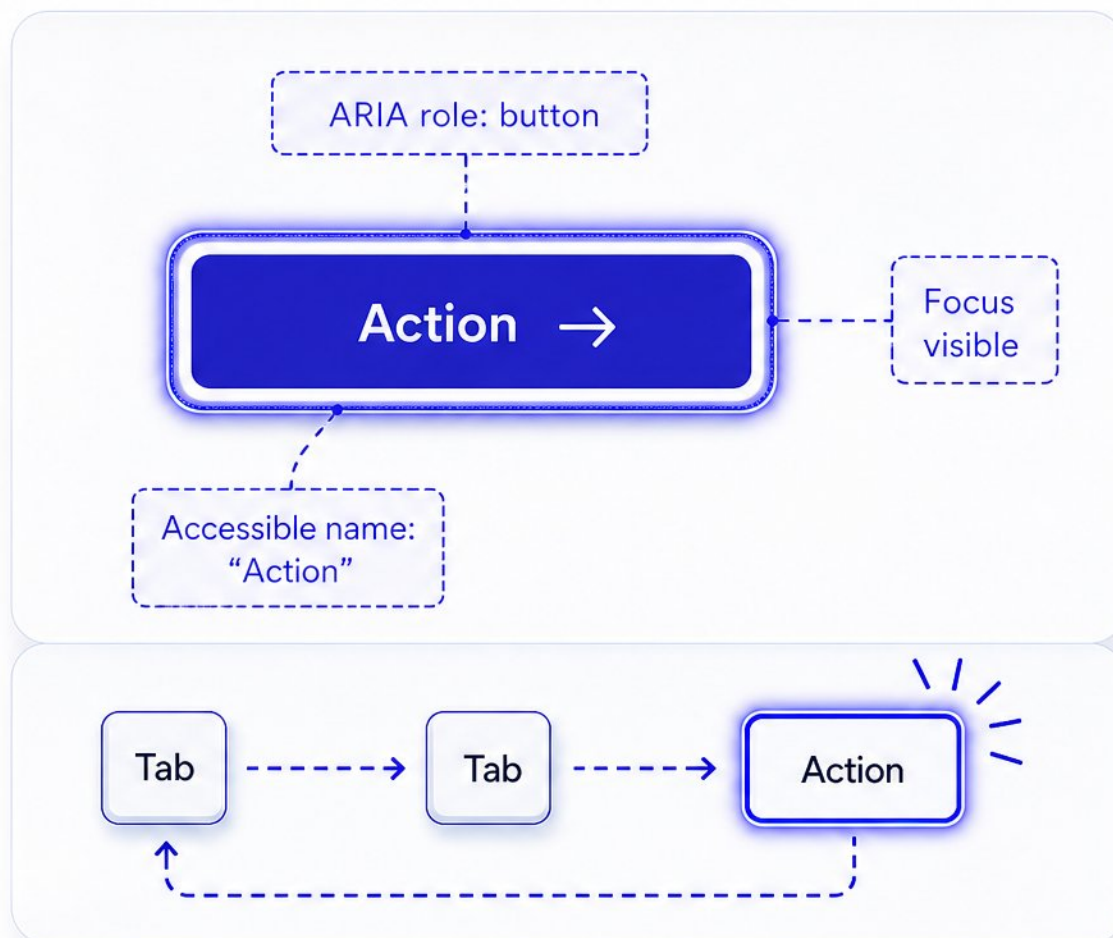
- Keyboard navigation, focus management, and ARIA roles



- Responsive behavior across breakpoints and input types



- Built-in accessibility is fixed once, benefits everywhere



Evaluating and Choosing a Framework



- Evaluate maturity, docs, customization, accessibility, and bundle size



- Build vs. buy vs. adapt: weigh team size and long-term needs



- Leading options in 2026: MUI, Ant Design, Fluent, shadcn/ui, Mantine



- Run a small pilot before committing to a full-scale rollout



Maturity



Docs



Customization



Accessibility



Bundle Size

MUI					
Ant Design					
Fluent					
shadcn/ui					
Mantine					

Common Pitfalls and How to Avoid Them



- Over-engineering: too many variants create complexity, not flexibility



- No named owner: systems drift without clear accountability



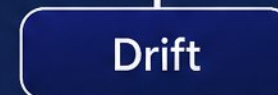
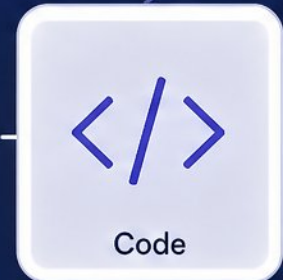
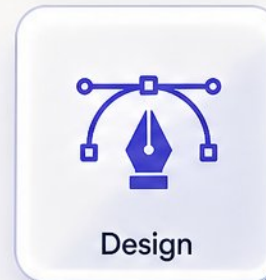
- Design-code mismatch: silent drift when values and components diverge



- Governance as a lightweight process, not a bureaucratic gate



- Treat local workarounds as feedback, not discipline failures



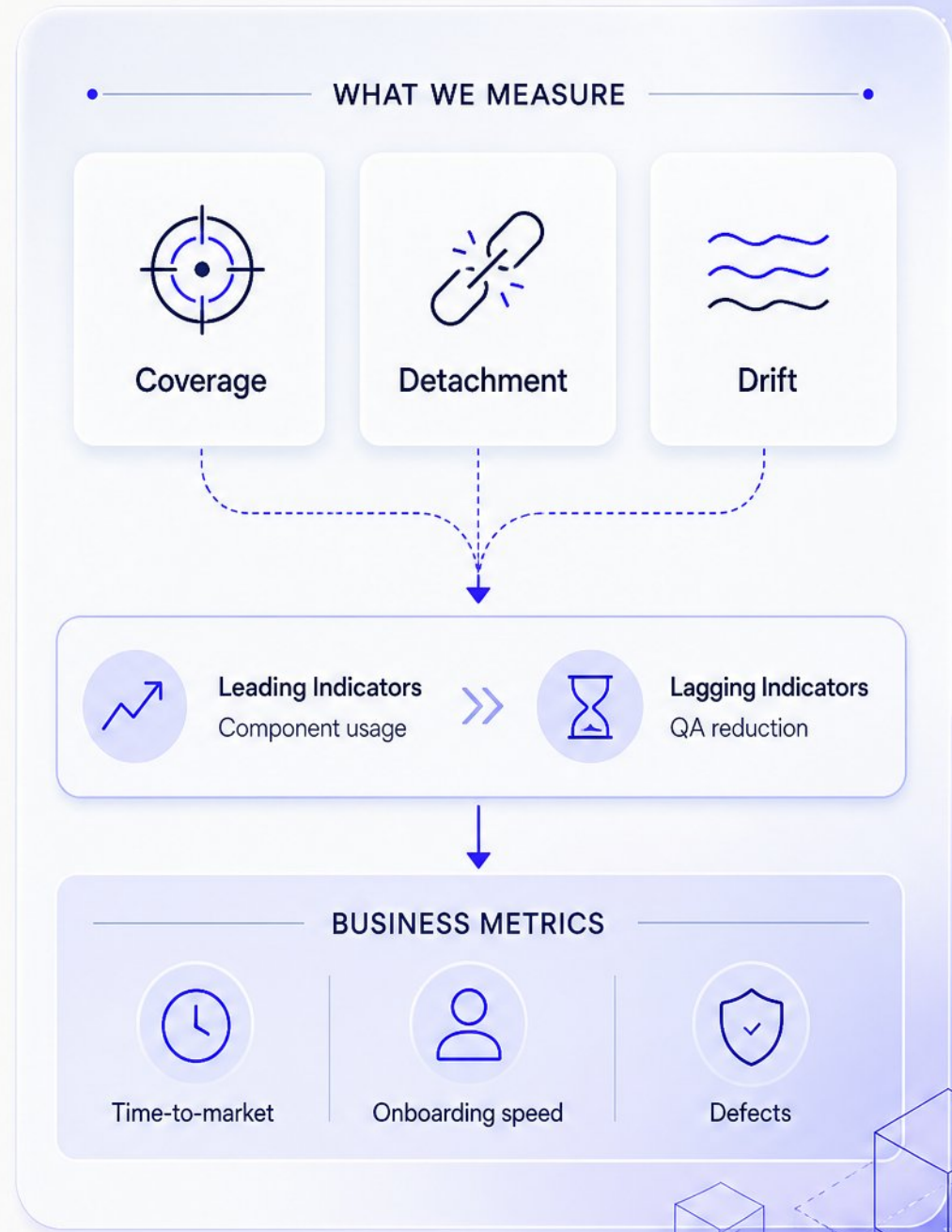
Adoption and Governance



- ✓ - Adoption is the only metric that matters at the program level
- ✓ - Clear contribution path: propose, review, and approve changes
- ✓ - Versioning, deprecation, and communication form governance foundations
- ✓ - Treat the design system like a product, not a one-time deliverable

Measuring What Matters

- Track usage, coverage, and detachment rates in Figma
- Watch custom-component drift as a triage signal
- Lead indicators: component usage; lagging: QA reduction
- Business metrics: time-to-market, onboarding speed, defects
- Run regular audits and close the feedback loop



Planning Your First Framework



- Audit current UI: identify pain points and duplicate components



- Define MVP scope: tokens first, then 5–10 high-frequency components



- Run a pilot on a small, representative project



- Establish feedback loops and iterate before scaling to more teams



Action Plan and Next Steps



- Steps: audit → MVP scope → pilot → iterate



- Name an owner, lay the token baseline, pick one pilot project



- Pilot with 5–10 high-frequency components on a small, representative project



- Review success metrics bi-weekly; define leading and lagging KPIs



- Extend scope and contribution paths once the first wins show



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/ae47980f/public