Cheat Sheet: Web Components vs React Components

■ Decision Flow

Do you want reusable UI across multiple frameworks?
|-- Yes → **Use Web Components ■**
- Design systems
- Micro-frontends
- Standalone widgets (chat, date picker, chart)

|-- No (App is in React only) → **Use React Components ■■**
- Dashboards
- SPAs
- Apps with complex state
- Need fast development

■ Quick Comparison Table

Criteria	Web Components ■	React Components ■■
Runs in	Any framework (React, Vue, Angular, plain HTML)	Only React
Encapsulation	Shadow DOM → true isolation	CSS-in-JS / modules
State handling	Manual, DOM-based	Hooks (useState, useEffect), Context, Redux
Best Use Case	Design systems, shared widgets, micro-frontends	Full React applications
Benefits	Cross-framework, future-proof, reusable	Easy state mgmt, rich ecosystem, fast DX
Performance	Native DOM (no VDOM overhead)	Virtual DOM (efficient diffing for big apps)

■ Quick Trick to Memorize

Web Components → Universal Building Blocks
(Think: “Make once, use anywhere”)

React Components → Application Lego Blocks
(Think: “Best inside React house”)