Alexandre Zajac • 2nd
SDE & AI @Amazon | Building Hungry Minds to 1M+ | Daily Posts on Software Engineering, Sy...
2h •

Follow

I spent my Sunday studying how React works so you don't have to.

React explained in 60 seconds:

React powers 15% of the web.

If you're serious about building modern web applications, you need to understand how it works.

0. Virtual DOM & reconciliation:

↳ Lightweight in-memory representation of the actual DOM

↳ Diffing algorithm calculates minimal updates (O(n^3) to O(n) heuristic)

↳ Batched updates to avoid unnecessary reflows

↳ Keys optimize list element tracking

1. Component architecture:

↳ Function components with hooks (useState, useEffect) manage state/lifecycle

↳ Class components with legacy lifecycle methods (componentDidMount, etc.)

↳ Hooks are closure-based state isolation with dependency arrays

↳ Custom Hooks are reusable, composable logic (e.g., useFetch, useMemory, etc.)

2. JSX & React elements:

↳ JSX transpiles to React.createElement(type, props, children)

↳ Elements are immutable descriptors, not instances

↳ ReactDOM renders elements into fiber trees during reconciliation

3. Fiber architecture:

↳ Rewrite of React's core algorithm (2017)

↳ Linked list of fibers: Each fiber represents a unit of work

↳ Time slicing: Pause/resume rendering via requestIdleCallback

↳ Priority levels: Concurrent mode schedules urgent vs. deferred updates

4. Concurrent mode:

↳ Suspense: Defer rendering until data/code-splitting resolves

↳ Transitions: Mark non-urgent state updates (startTransition)

↳ Streaming SSR: Send HTML chunks progressively with React 18

5. State management:

↳ useState: Uses dispatcher and queue behind the scenes

↳ Context API: Prop drilling alternative with Provider/Consumer

↳ Redux Integration: Middleware (thunks, sagas) for side effects

↳ My favorite is still React Query or Zustand for almost any app

6. Performance optimizations:

↳ Memoization: React.memo, useMemo, useCallback to prevent re-renders

↳ Lazy Loading: React.lazy + Suspense for code splitting

↳ Profiler API: Measure component render times in dev tools

7. Ecosystem & extensibility:

↳ Next.js: SSR/SSG/ISR, file-based routing, API routes

↳ React Router: Declarative routing with nested layouts

↳ Testing: React Testing Library, Jest integration

↳ Native: React Native bridges to native UI threads

React gives you a declarative, component-driven model with deterministic rendering.

Its balance of abstraction and control is outstanding, you can drop to imperative code when needed.

What’s your go-to frontend framework, and why?

#softwareengineering #react #programming