React Explained

In 60 Seconds

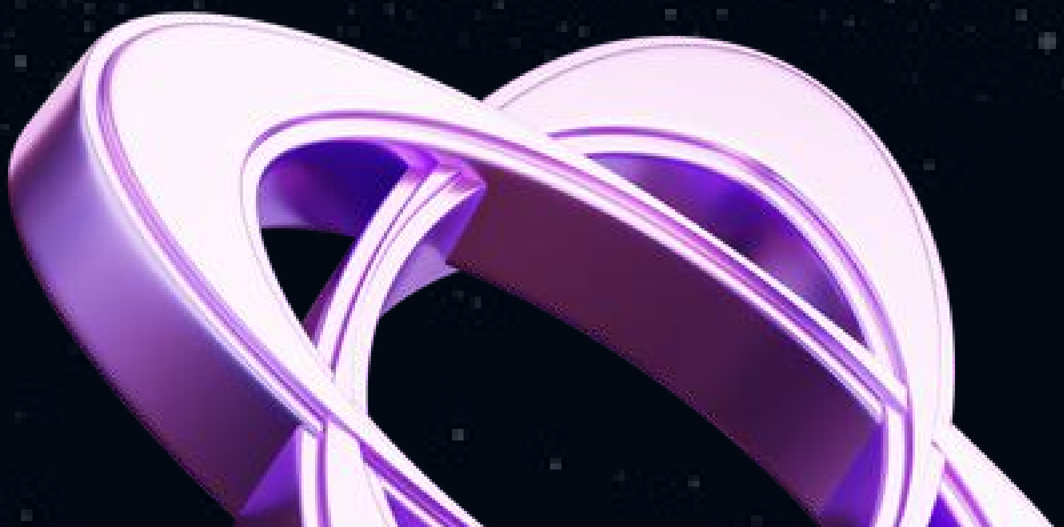
0. Virtual DOM & reconciliation:

Lightweight in-memory representation of the actual DOM.

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

Batched updates avoid unnecessary reflows.

Keys optimize list element tracking.



Swipe

1. Component Architecture:

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

Class components use legacy lifecycle methods (`componentDidMount`, etc.).

Hooks are closure-based state isolation with dependency arrays.

Custom Hooks enable reusable, composable logic (e.g., `useFetch`).

Swipe

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.

Swipe

3. Fiber Architecture:

Rewrite of React's core algorithm (2017).

Linked list of fibers where each represents a unit of work.

Time slicing enables pause/resume rendering via `requestIdleCallback`.

Priority levels schedule urgent vs deferred updates in Concurrent Mode.

Swipe

4. Concurrent Mode:

Suspense defers rendering until data/code-splitting resolves.

Transitions mark non-urgent state updates (`startTransition`).

Streaming SSR sends HTML chunks progressively (React 18+).

Swipe

5. State Management:

`useState` uses a dispatcher and queue behind the scenes.

Context API avoids prop drilling with Provider/Consumer.

Redux middleware (thunks/sagas) handles side effects.

`React Query/Zustand` simplify data/state for most apps.

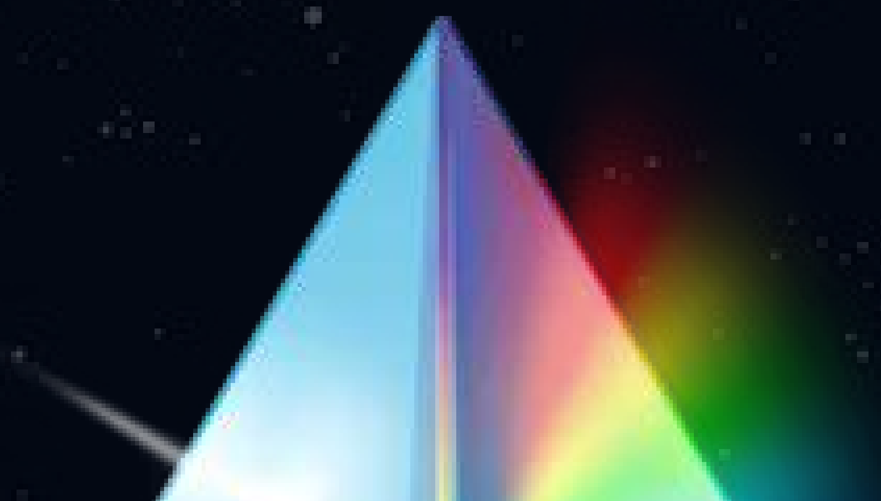
Swipe

6. Performance Optimizations:

Memoization via `React.memo`, `useMemo`, `useCallback` prevents re-renders.

Lazy Loading with `React.lazy` + Suspense enables code splitting.

Profiler API measures render times in dev tools.



Swipe

7. Ecosystem & Extensibility:

`Next.js` offers SSR/SSG/ISR, file-based routing, and API routes.

`React Router` enables declarative routing with nested layouts.

Testing via React Testing Library + `Jest` integration.

`React Native` bridges to native UI threads.

Swipe

If You Found This Valuable

Like
Comment
Repost
Save