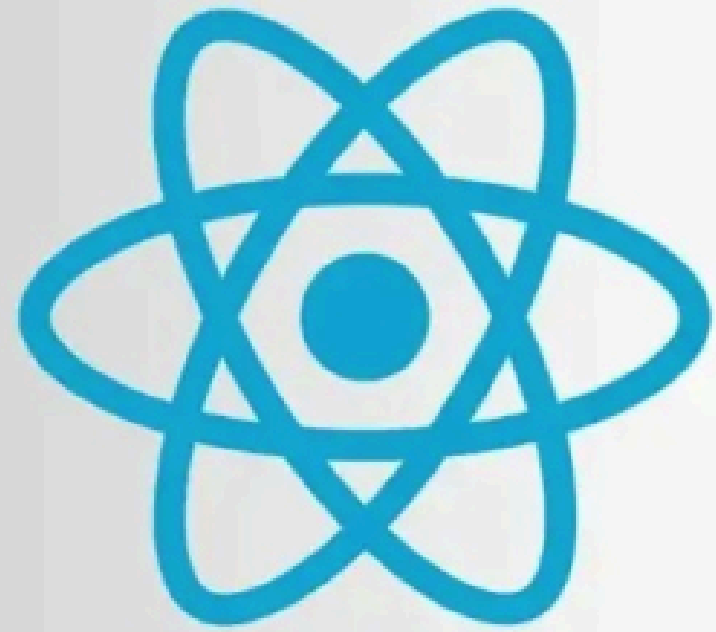
ADVANCED

REACT.JS



IN THIS CAROUSEL,
WE'LL EXPLORE ADVANCED REACT.JS
CONCEPTS

Asad Saeed

1. React Server Components

Run components on the server to send pre-rendered HTML, reducing client-side JS and boosting performance. Ideal for data-heavy apps like blogs or e-commerce!

```jsx
// ServerComponent.jsx
export default async function ServerComponent() {
  const data = await db.query('SELECT * FROM products');
  return <div>{data.name}</div>;
}
// Client: App.jsx
<Suspense fallback={<Spinner />}>
  <ServerComponent />
</Suspense>
```

- Faster initial loads & better SEO.
- Shrinks client-side bundle size.
- Stable in React 19 (2025).
- Perfect for content-driven sites.

Asad Saeed

2. Concurrent Rendering

React prioritizes urgent UI updates, making apps feel snappy even during heavy computations. Use it for real-time search or interactive dashboards!

```jsx
function SearchBar() {
  const [query, setQuery] = useState('');
  const [isPending, startTransition] = useTransition();
  const handleInput = (e) => {
    startTransition(() => setQuery(e.target.value));
  };
  return <input onChange={handleInput} placeholder="Type here..." />;
}
```

- Smooth UI with non-blocking updates.
- Default in React 19.
- useTransition for heavy tasks.
- Ideal for real-time apps.

Asad Saeed

3. Suspense for Data Fetching

Simplifies async data handling with automatic loading states. Perfect for fetching API data in e-commerce or social media apps!

```jsx
const LazyData = React.lazy(() =>
  fetch('/api/posts').then(res => ({
    default: () => <div>{res.data}</div>
  })));
function App() {
  return (
    <Suspense fallback={<Spinner />}>
      <LazyData />
    </Suspense>
  );
}
```

- Clean loading states with fallbacks.
- Streams data in React 19.
- Simplifies API integrations.
- Pairs with Server Components.

Asad Saeed

4. Automatic Batching

Groups multiple state updates into one render for better performance. Use it in forms or interactive UIs to avoid unnecessary re-renders!

```
function Form() {
  const [name, setName] = useState('');
  const [age, setAge] = useState(0);
  const handleSubmit = () => {
    setName('John');
    setAge(30); // Batched in React 19
  };
  return <button onClick={handleSubmit}>Submit</button>;
}
```

- Fewer renders for faster apps.
- Works in events/promises (React 19).
- Optimizes complex UIs.
- No manual batching needed.

Asad Saeed

5. useSyncExternalStore Hook

Syncs external data (e.g., browser APIs, Redux) with React's rendering, preventing issues in concurrent mode. Great for custom hooks!

```
function useWindowSize() {
  return useSyncExternalStore(
    (cb) => {
      window.addEventListener('resize', cb);
      return () => window.removeEventListener('resize', cb);
    },
    () => window.innerWidth
  );
}
function App() {
  const width = useWindowSize();
  return <div>Width: {width}px</div>;
}
```

- Syncs external stores safely.
- Prevents tearing in concurrent mode.
- Supports SSR with defaults.
- Ideal for custom hooks.

Asad Saeed

6. React Compiler (React Forget)

Automatically memoizes components and hooks to optimize performance without manual useMemo or useCallback. Boosts app speed effortlessly!

```
// No manual memoization needed!
function ExpensiveComponent({ data }) {
  const result = heavyComputation(data); // Auto-optimized by React Compiler
  return <div>{result}</div>;
}
// React Compiler handles memoization internally
```

- Auto-optimizes expensive computations.
- Experimental in 2025, likely in React 19.
- Reduces boilerplate (useMemo, etc.).
- Great for performance-critical apps.

Asad Saeed

7. useOptimistic Hook

Updates UI optimistically before server confirmation, making apps feel instant. Perfect for likes, comments, or form submissions!

```jsx
function LikeButton({ postId }) {
  const [optimisticLikes, addOptimisticLike] = useOptimistic(0);
  const handleLike = async () => {
    addOptimisticLike(likes => likes + 1);
    await api.likePost(postId);
  };
  return <button onClick={handleLike}>{optimisticLikes} Likes</button>;
}
```

- Instant UI updates for better UX.
- New in React 19 (2025).
- Simplifies optimistic updates.
- ideal for social media apps.