🚀React Batching — Complete Senior-Level Notes

What is Batching in React?

Batching = grouping multiple state updates into a single render to avoid unnecessary re-renders.

- React doesn't update the DOM immediately every time you call `setState`.
- Instead, it collects multiple updates and processes them together.
- Reduces DOM operations, which are expensive, and improves performance.

Simple Analogy:

- Imagine writing on a whiteboard.
- **Without batching** → every word triggers a clean and rewrite.
- **With batching** → you write several words first, then the board is updated once.
- Saves effort and improves speed.

Why Batching is Important

Performance Optimization:

- DOM updates are costly. Fewer re-renders → faster UI.

Consistency:

- Multiple state updates happen together → avoids intermediate inconsistent UI states.

React Fiber & Concurrency:

- Fiber allows React to schedule, pause, and batch updates efficiently for smooth UI.

How Batching Works

3.1 In Class Components

```
class Counter extends React.Component {
  state = { count: 0 };

  incrementTwice = () => {
    this.setState({ count: this.state.count + 1 });
    this.setState({ count: this.state.count + 1 });
  };
```

```
  render() {
    console.log('Render called');
    return <button onClick={this.incrementTwice}>{this.state.count}</button>;
  }
}
```

- Updates inside **event handlers are automatically batched**. - Count increases only once per render despite two `setState` calls.

3.2 In Functional Components

```
function Counter() {
  const [count, setCount] = React.useState(0);

  const incrementTwice = () => {
    setCount(count + 1);
    setCount(count + 1);
  };

  console.log('Render called');
  return <button onClick={incrementTwice}>{count}</button>;
}
```

- React 18+ automatically batches multiple `setState` calls inside event handlers.

React 18 Automatic Batching

- **Before React 18:** Only React event handlers batched updates.
- **React 18+:** Automatic batching everywhere, including:
- Promises
- setTimeout
- Native events
- Async operations

Example:

```
function App() {
  const [count, setCount] = React.useState(0);
  const [text, setText] = React.useState("");

  const handleClick = () => {
    setCount(c => c + 1);
    setText("Updated");
    console.log("Inside handleClick");
  };
```

```
  return <button onClick={handleClick}>{count} - {text}</button>;
}
```

- Both state updates are **batched → only one render happens**.

Explicit Batching

- Use `flushSync` to **force updates immediately**, bypassing batching.

```
import { flushSync } from 'react-dom';

flushSync(() => {
  setCount(c => c + 1);
  setText("Updated immediately");
});
```

- Useful when **DOM must update synchronously**.

How Batching Helps Performance

Without batching:

```
setCount(count + 1) → Render → DOM update
setText("Hi") → Render → DOM update
```

With batching:

```
setCount(count + 1)
setText("Hi") → Single Render → Single DOM update
```

- Reduces layout calculations, reflows, repaints - Makes UI smoother

---

Batching and React Fiber

- Fiber enables batching efficiently.
- React schedules updates in **units of work**.
- Can pause low-priority updates while processing high-priority ones.
- Combines multiple updates into **one commit → smoother UI**.

Key Points / Rules

- Automatic batching happens for:
- React event handlers
- React 18+ async code (setTimeout, promises)
- Before React 18, async updates cause multiple renders.
- `flushSync` can break batching for immediate DOM updates.

Practical Example: React 18 vs React <18

```
setTimeout(() => {
  setCount(c => c + 1);
  setText("Hi");
}, 0);
```

- React 17 → Two renders (no batching) - React 18 → One render (automatic batching)

🚚Summary Table

Concept	Before React 18	React 18+	Notes
Event Handler	Batched	Batched	✅
setTimeout / Promise	Not batched	Batched	✅Automatic batching
flushSync	N/A	Forces immediate	⚠️Breaks batching
Benefit	Some performance	Full performance	Reduced renders, smoother UI

Key Takeaways

- Batching groups multiple state updates into one render → improves performance.
- React 18+ automatically batches everything, even async updates.
- Fiber tracks work and schedules updates efficiently.
- `flushSync` is used for synchronous updates when necessary.
- Always rely on batching to reduce unnecessary DOM operations and keep UI smooth.

Visualization Diagram

Refer to the attached diagram showing: - React 17 batching (only event handlers) - React 18 automatic batching (includes async updates) - Current Fiber → WIP Fiber → commit phase flow - Queued updates processed in one render for performance


React Batching Diagram