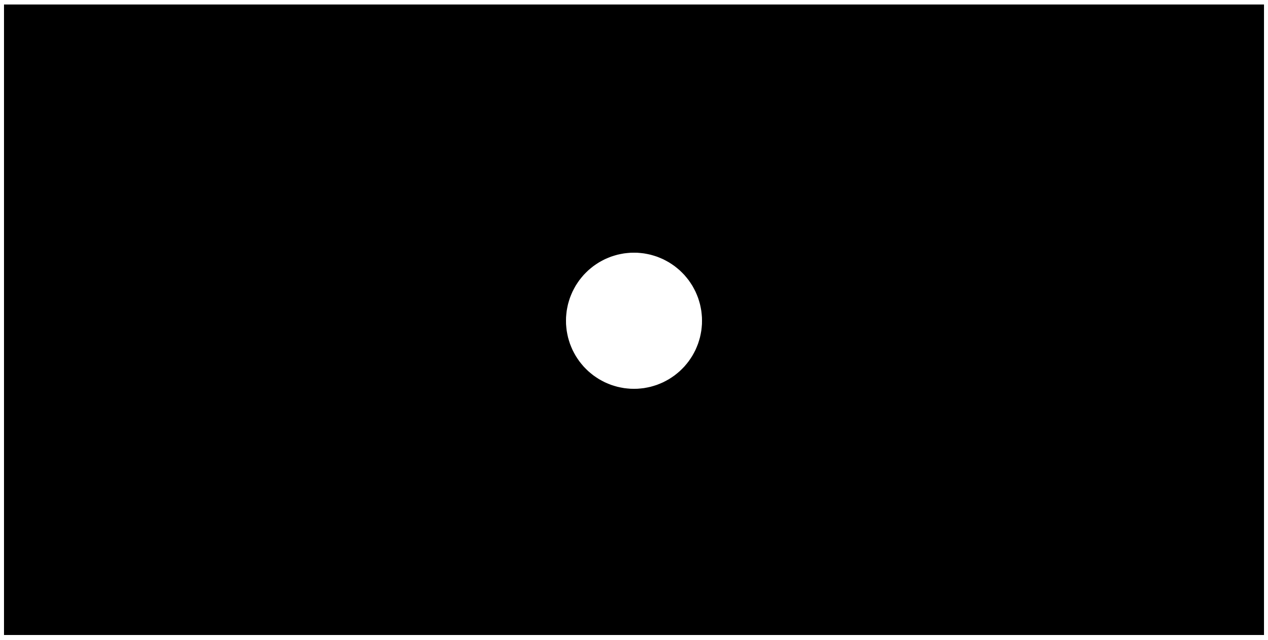
Alex Sidorenko

Optimizing React performance without refs and memo

January 04, 2023

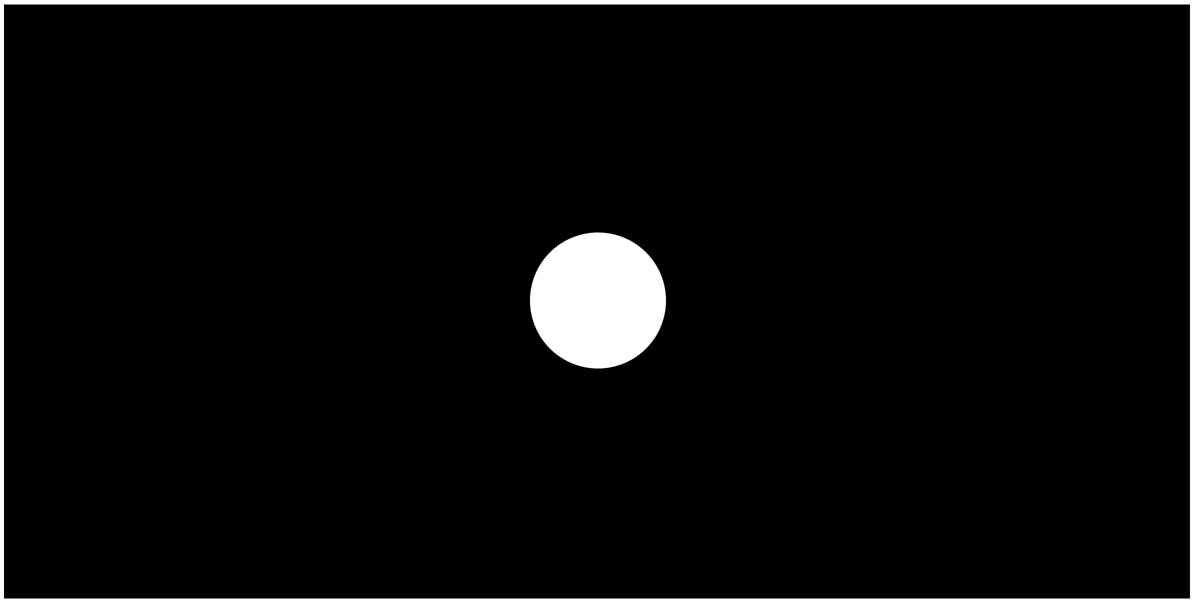
Frequent state updates in React can lead to performance problems. For example, this component shows the current mouse position while rendering an expensive JSX.



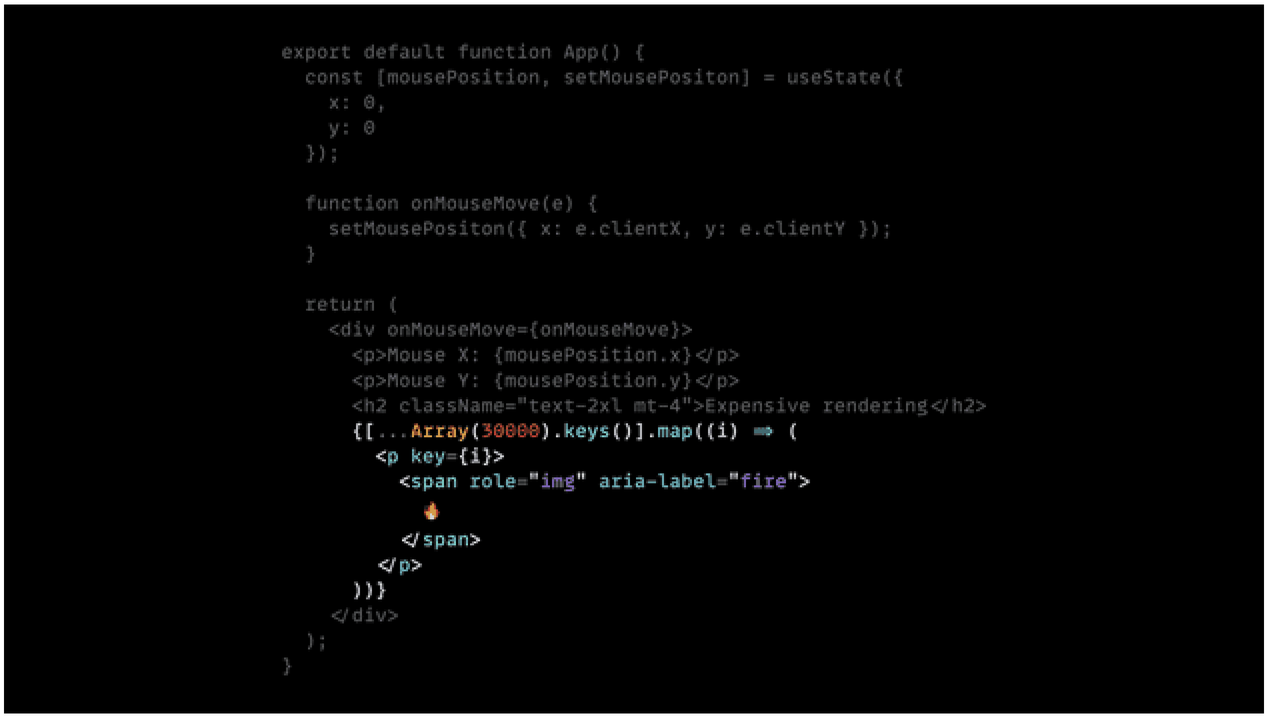
How can we make it faster? Should we find a way to stop re-rendering the component that often? Is it a case for refs? Should we apply memoization with `memo` or `useMemo` ? **There is another solution ✨**

But first, why exactly is it slow?

React is smart enough to update only the minimum necessary parts of the DOM. If we check the DOM inspector, we'll see that the only things that get updated are the mouse coordinates.



And yet, the app is still slow. That's because every render returns a tree of elements that React has to compare with a tree from the previous render to determine whether to change the DOM (Reconciliation). And we generate a lot of elements in the component that is being re-rendered on every mouse move.

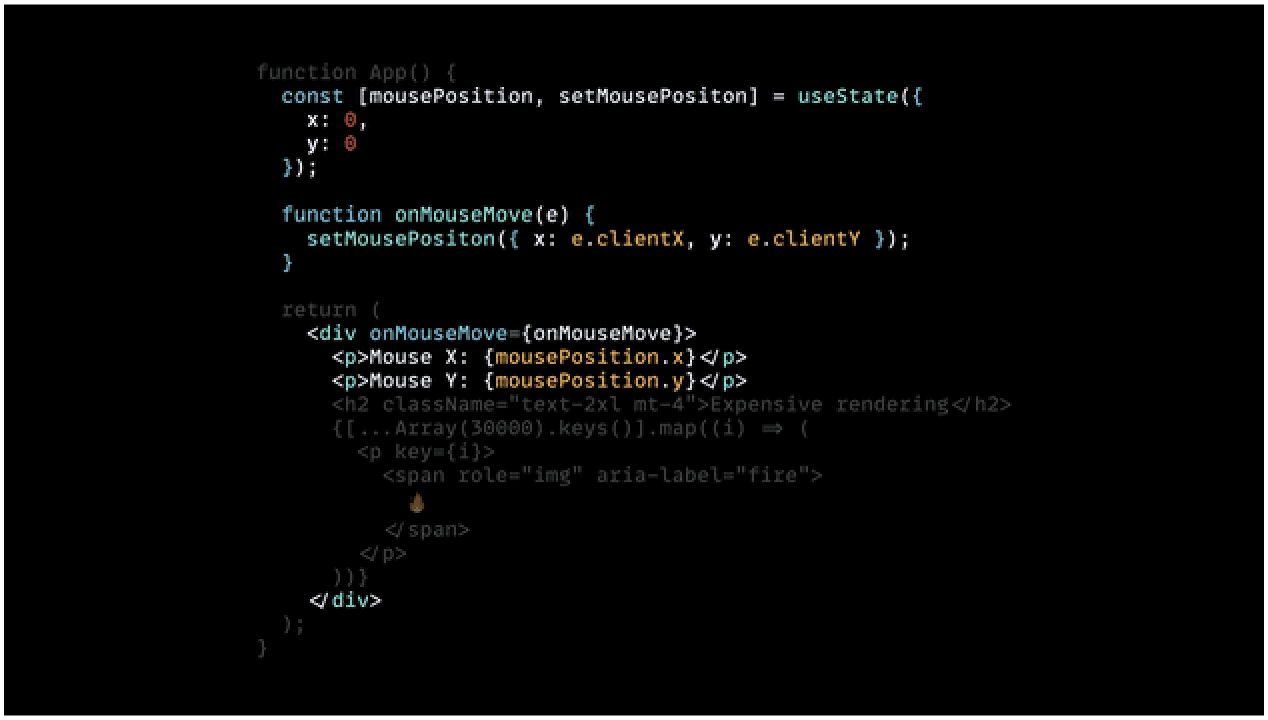


```
export default function App() {
  const [mousePosition, setMousePositon] = useState({
    x: 0,
    y: 0
  });

  function onMouseMove(e) {
    setMousePositon({ x: e.clientX, y: e.clientY });
  }

  return (
    <div onMouseMove={onMouseMove}>
      <p>Mouse X: {mousePosition.x}</p>
      <p>Mouse Y: {mousePosition.y}</p>
      <h2 className="text-2xl mt-4">Expensive rendering</h2>
      {[...Array(30000).keys()].map((i) => (
        <p key={i}>
          <span role="img" aria-label="fire">
            🔥
          </span>
        </p>
      ))}
    </div>
  );
}
```

State colocation

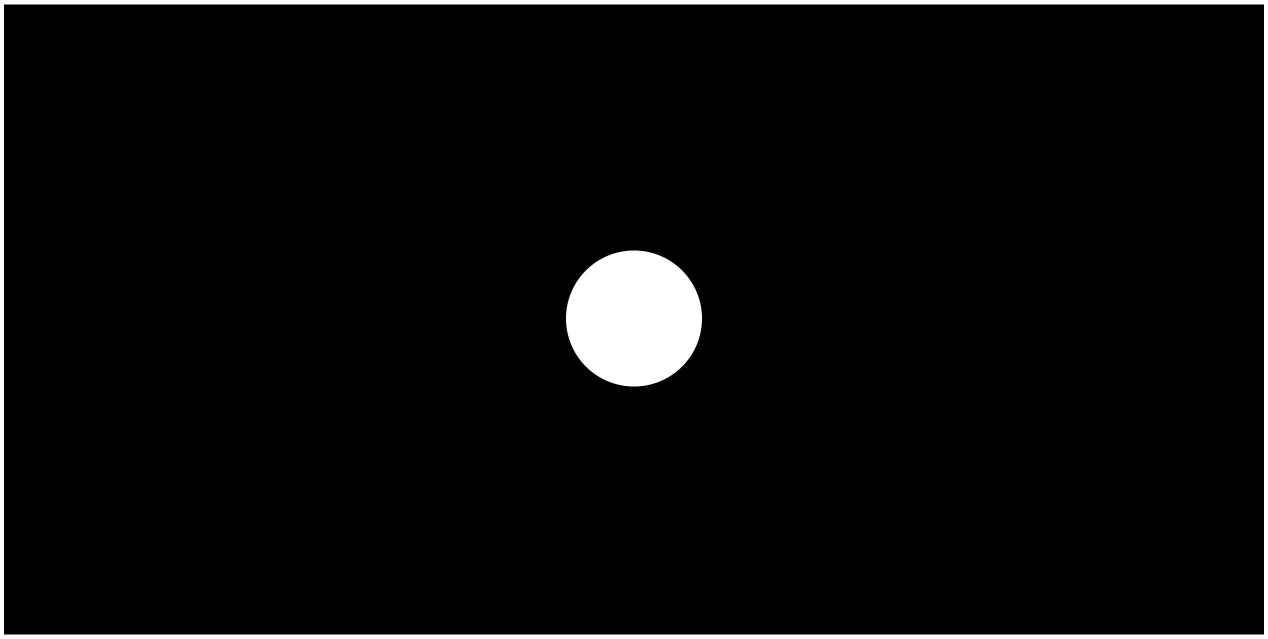
The simplest way to avoid this type of performance problems in React is to colocate the state with the parts of UI this state is updating.

In our case, the parts of our component related to the mousemove updates are 👇

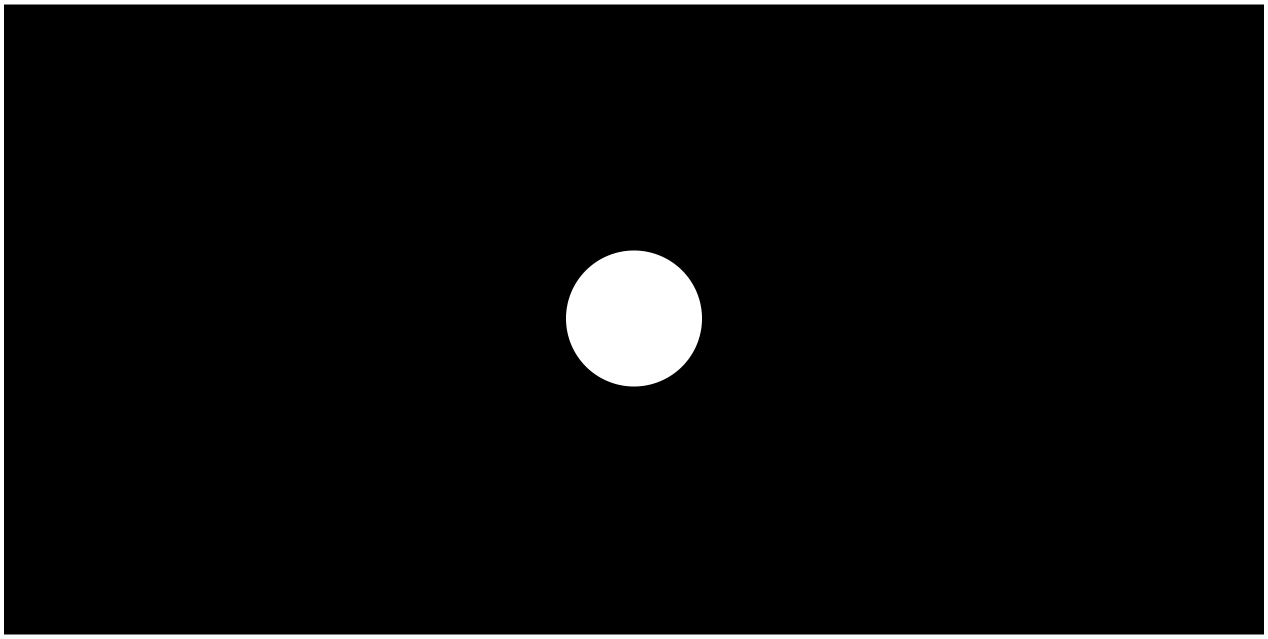


```
function App() {
  const [mousePosition, setMousePositon] = useState({
    x: 0,
    y: 0
  });

  function onMouseMove(e) {
    setMousePositon({ x: e.clientX, y: e.clientY });
  }

  return (
    <div onMouseMove={onMouseMove}>
      <p>Mouse X: {mousePosition.x}</p>
      <p>Mouse Y: {mousePosition.y}</p>
      <h2 className="text-2xl mt-4">Expensive rendering</h2>
      {[...Array(30000).keys()].map((i) => (
        <p key={i}>
          <span role="img" aria-label="fire">
            🔥
          </span>
        </p>
      ))}
    </div>
  );
}
```

Let's extract these parts into a separate component.



Now, let's add `children` prop to our newly created component and pass the rest of JSX there.



That's it 🎊. We isolated mousemove state updates within the component responsible only for those updates. Now our app is faster since it doesn't need to re-render the expensive part of JSX on every mouse move. And all we did was move components around. No refs, no memoization.

Coding Challenge 🙌

The app in this CodeSandbox shows a current scroll position by re-rendering the entire component on the scroll event, and it's not very performant. Could you make it faster without using refs and memo?

Hint: you may not need a `children` *prop for this one*

Send the link to your solution as the reply to this tweet.

Related articles

- Before you memo by Dan Abramov
- State Colocation will make your React app faster by Kent C. Dodds

Don't have time for long boring React articles?

I write short, to the point, visual posts about React. Subscribe to get notified when a new one comes out.

First Name

Email Address

Subscribe

I respect your privacy. Unsubscribe at any time.

← Why does onClick={x()} causes "too many re-renders" error in React?