Reducing bugs in React codebase

Understanding anti-patterns in React

Darshita Chaturvedi

Co-Founder & CEO, Atri Labs

React Native EU

September 1, 2022

About Me

- **Maintainer of Atri engine, and Co-Founder & CEO of Atri Labs**

  Creating a new full-stack web development framework

- **Previously, Graduate Research Assistant at MIT and Quant Researcher at BlackRock**

  Experiences in academia and quantitative finance industry

- **Editor of novels written by my grandfather over the last 50 years**

  Favorite past times: painting and learning art history

Motivation

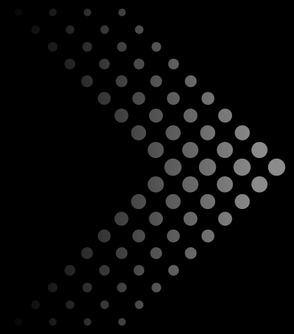
Many interactions in one page

Suboptimal code in one interaction can ruin the entire application

Code quality is paramount to support building production-grade applications

Requirement to build rigorous internal standards on code quality

Types of bugs



- Component does not update upon a user event
- Component updates partially upon a user event
- Component renders unexpectedly

Our origin story

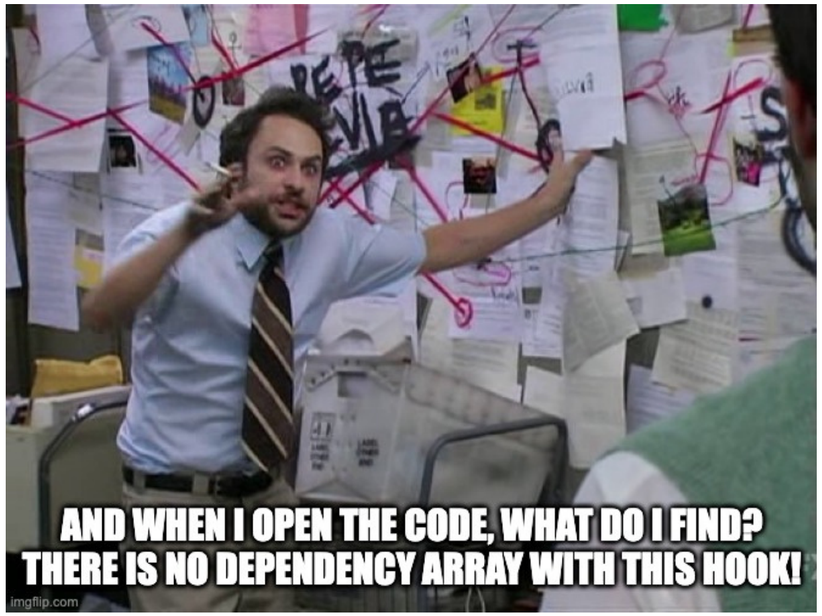
React code should be independent of:

- sequence in which components will render
- when components will render.

Hints

1 Hook without a dependency array

- In React, different pieces of code are linked to each other by dependencies.
- These pieces of interdependent code together keep the application state in its desired form.
- Therefore, if we use a hook that does not have a dependency, there is a high chance that it will result in bugs.



Be cautious while using hooks that do not take dependency arrays

What is a Pattern?

We characterize a React code as a good pattern if:

1. **Component is reusable**
2. **Code is easier to review and debug**

! Code is still considered a pattern if we wrote more lines of code or we (expectedly) introduced a few extra renders in order to achieve the above objectives.

Sandbox

Moving away from useState

```tsx
const Slider: React.FC<SliderComponentTypes> = ({
  minValue,
  maxValue,
  givenValue,
}) => {
  const [startPosition, setStartPosition] = useState<{…
  } | null>(null);
  const [finishPosition, setFinishPosition] = useState({ x: 0, y: 0 });
  const [currentPosition, setCurrentPosition] = useState({ x: 0, y: 0 });
  const [thumbPosition, setThumbPosition] = useState("0%");
  const [value, setValue] = useState(0);

  function getPosition(e: React.MouseEvent) {…
  }

  const handleMouseMove = useCallback(
    (e: any) => {
      const { pageX, pageY } = getPosition(e);
      console.log(pageX, pageY);
      setCurrentPosition({ ...currentPosition, x: pageX, y: pageY });
      console.log("current posi", currentPosition);
    },
    [currentPosition]
  );
  const handleThumbMouseUp = useCallback(…
  const handleThumbMouseDown = useCallback(…

  useEffect(() => {…
```

```tsx
const Slider: React.FC<SliderProps> = (props) => {
  const onMouseDown = useCallback(
    (e: React.MouseEvent) => {
      const startPostion = { x: e.pageX, y: e.pageY };
      const onMouseMove = (e: MouseEvent) => {
        if (startPostion) {
          const delta = e.pageX - startPostion.x;
          props.onChange(props.value + delta);
        }
      };
      const onMouseUp = () => {
        // unsubscribe
        document.removeEventListener("mousemove", onMouseMove)
        document.removeEventListener("mouseup", onMouseUp);
      };
      // subscribe
      document.addEventListener("mousemove", onMouseMove);
      document.addEventListener("mouseup", onMouseUp);
    },
    [props]
  );
```

Atri framework

0.35

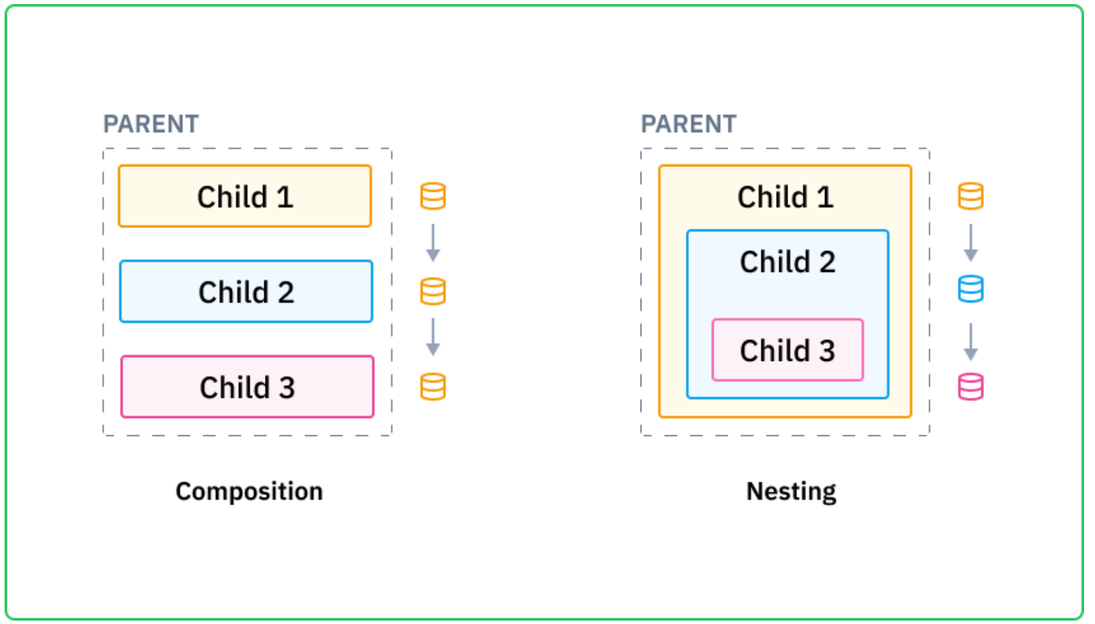
$$\frac{\#\ \text{useState}}{\#\ (\text{custom hooks} + \text{components})}$$

No. of components = 192
No. of custom hooks = 112
No. of useState in components = 55
No. of useState in custom hooks = 50

Hints

2 Nesting can lead to duplication of states

- While using nesting, we often provide duplicate states to the children. This can introduce bugs.
- Moreover, if we observe that there is a bug in *Child 3*, the time complexity of debugging in nesting is O (n) where n is the number of children above *Child 3*. By contrast, the time complexity of debugging in composition is O (1).



Avoid nesting since it may introduce bugs and make debugging harder.

Key Takeaways

Hooks without dependency array
Nesting approach to arrange components

props or context as initial state
Destroy and Recreate

Internal state in JSX
props as a dependency in useMemo (recommended)

React code should be independent of:

- **sequence in which components will render**
- **when components will render.**

Thank you for listening!

Darshita Chaturvedi

@darshitac_