

React API

Кирилл Талецкий

TeachMeSkills
2 Октября 2023

Комбинируем хуки

Задача: компонент для обратного отсчёта

- Нам нужен компонент, который умеет делать обратный отсчёт
- Такой может пригодиться для форм, в которых нужно ограничить время повторных запросов:
 - Сообщения в групповых чатах
 - Сброс пароля

Forgot Password?

Please enter the email address or username associated with your account. We'll send you a link to reset your password and get you back into your account.

kirill.t+4@getquin.com

Cancel

Resend in 29

Переиспользуемый хук для обратного отсчёта

```
interface UseCountdownState {
  isOver: boolean;
  count: number;
}
interface UseCountdownHelpers {
  start: () => void;
}
type UseCountdownApi = [UseCountdownState, UseCountdownHelpers];

export const useCountdown = (initialCount = 60): UseCountdownApi => {
  const [count, setCount] = useState(initialCount);
  const [isOver, setIsOver] = useState(true);

  const timerRef = useRef<ReturnType<typeof setInterval>>();

  const start = useCallback(() => {
    setIsOver(false);
    setCount(initialCount);

    timerRef.current = setInterval(() => {
      setCount((prevState) => {
        if (prevState === 0) {
          clearInterval(timerRef.current);
          setIsOver(true);
          return 0;
        }

        return prevState - 1;
      });
    }, 1000);
  }, [initialCount]);

  useEffect(() => () => clearInterval(timerRef.current), []);

  return [{ isOver, count }, { start }];
}
```



React API



Раздаём данные всем компонентам

Context API

Задача

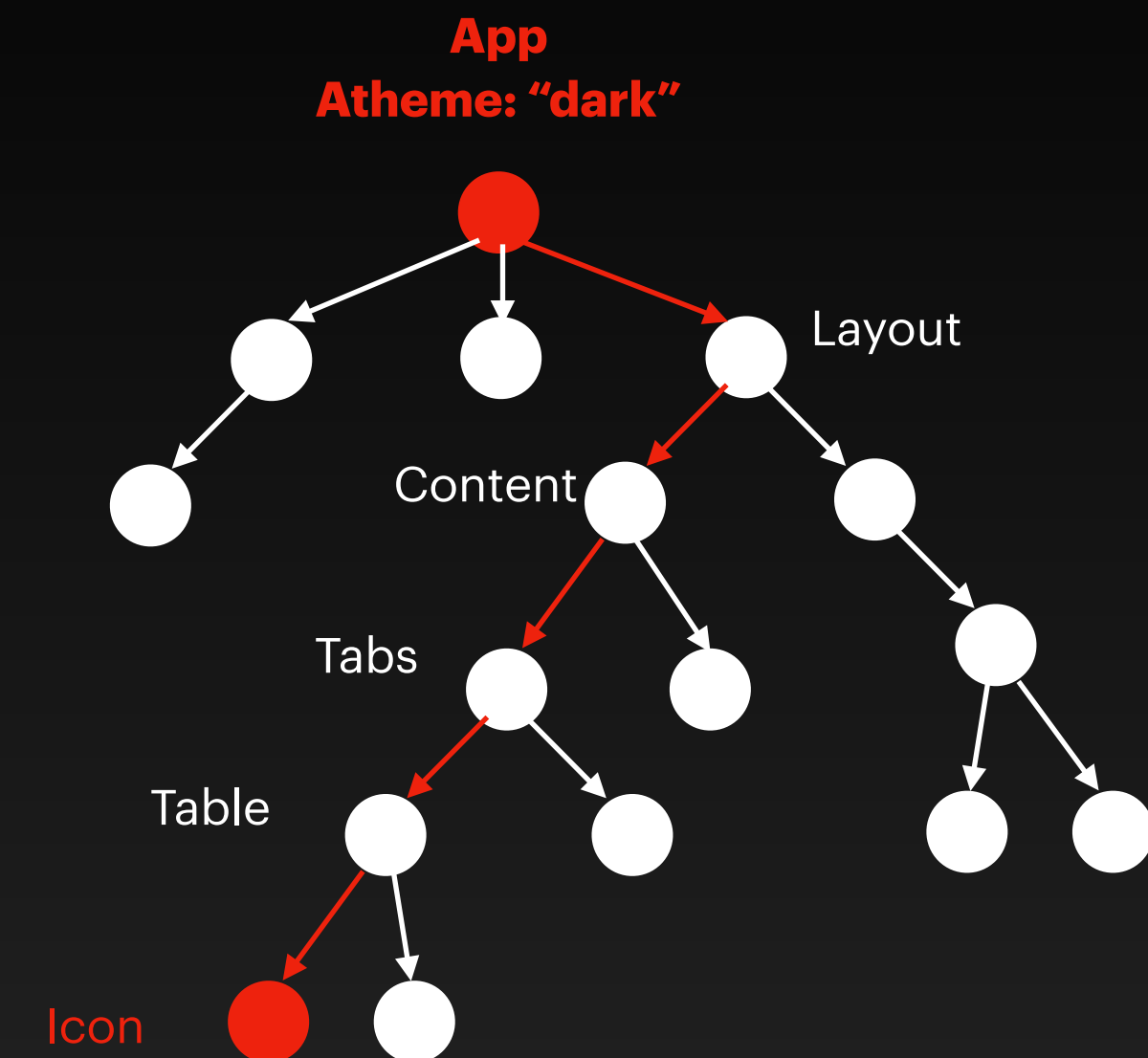
- Задача: вашу команду разработки попросили добавить тёмную тему в приложение
 - Вы разработали компонент, который умеет переключать тему
 - Цвета переключаются через чистый CSS
- Но у вас есть компоненты, которые отображаются по-разному, в зависимости от применённой темы
 - PNG иконки
 - JPEG фоны и картинки с промо

	Apple AAPL x1	€173.51 €173.51	€162.29 €162.29	+€0.50 ↗ 0.31%	⋮
	Ethereum ETH x0.5	€776.33 €1,552.67	● €793.81 €1,587.61	+€3.04 ↗ 0.38%	⋮

	Apple AAPL x1	€173.51 €173.51	€162.29 €162.29	+€0.50 ↗ 0.31%	⋮
	Ethereum ETH x0.5	€776.33 €1,552.67	● €793.81 €1,587.62	+€3.04 ↗ 0.38%	⋮

Задача

- Компоненты, зависящие от темы могут быть очень глубоко в дереве
- **Как передать в них информацию о теме?**
- Пропсы точно не подойдут:
 - вы не хотите, чтобы при изменении темы перерисовывались все компоненты в цепочке
 - К тому же, остальным компонентам в 99% случаев не нужно знать, что такое тема - их элементы меняют цвет через CSS



getquin

Q Search for stocks, etfs or i

Upgrade

Positions

+ Add transaction

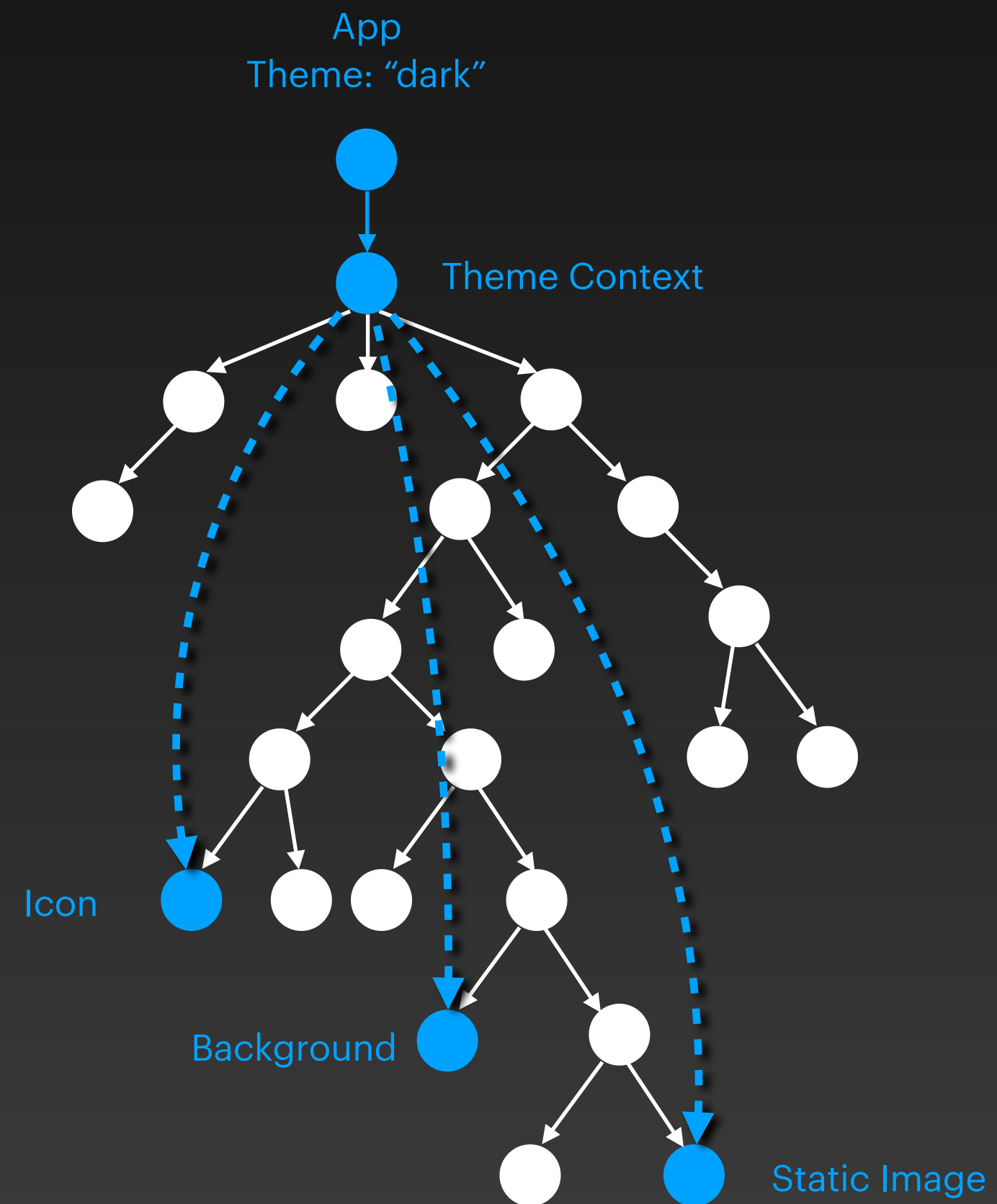
Alltime Intraday

Title	Buy in ↓	Position ↑	P/L ↓	
<div></div> <div>Procter & Gamble</div> <div>PG x60</div>	€541.38 €9.023	€8,278.80 €137.98	-€48.00 ↘ -0.58%	⋮
<div></div> <div>Bitcoin</div> <div>BTC x0.111</div>	€4,526.24 €40,776.95	€2,850.62 €25,681.29	+€13.04 ↗ 0.46%	⋮

React контекст

Приходит на помощь

- Контекст передаёт данные компонентам минуя пропсы



Исходное приложение

```
const Logo = () => {  
  // нам нужно текущее значение темы  
  // например, чтобы загрузить правильную картинку с бэкенда  
  //  
  // здесь для примера мы просто отобразим текущее значение темы  
  return <span>logo</span>;  
};
```

```
const List = () => {  
  return (  
    <ul>  
      <li><Logo /> item 1</li>  
      <li><Logo /> item 2</li>  
      <li><Logo /> item 3</li>  
    </ul>  
  );  
};
```

```
export const App = () => {  
  return (  
    <>  
      <header>  
        <h1>My App</h1>  
      </header>  
      <main>  
        <ThemeSwitcher />  
        <List />  
      </main>  
    </>  
  );  
};
```

Создаём контекст

```
const ThemeContext = React.createContext<ThemeContextData>({  
  colorScheme: "light",  
  setColorScheme: () => {},  
});
```

Начальное значение

Будет использовано в том случае, если значение из контекста попытаться забрать за его пределами

Оборачиваем приложение

Заводим состояние с текущей темой

```
const [colorScheme, setColorScheme] = useState<ColorScheme>("light");

return (
  <ThemeContext.Provider value={{ colorScheme, setColorScheme }}>
    <header>
      <h1>My App</h1>
    </header>
    <main>
      <ThemeSwitcher />
      <List />
    </main>
  </ThemeContext.Provider>
);
```

Передаём в него значение, которое будет передано во все компоненты-потребители

Забираем специальный компонент-провайдер из созданного контекста, оборачиваем нужную часть приложения

Потребляем контекст в компонентах

```
const ThemeSwitcher = () => {  
  // забираем значение текущей темы и метод для её переключения  
  // используем в чекбоксе  
  const { colorScheme, setColorScheme } = useContext(ThemeContext);  
  return (  
    <label>  
      <input  
        type="checkbox"  
        defaultChecked={colorScheme === "dark"}  
        onChange={(e) => {  
          setColorScheme(e.target.checked ? "dark" : "light");  
        }}  
      />  
      Dark mode  
    </label>  
  );  
};  
  
const Logo = () => {  
  // забираем значение текущей темы в логотипе  
  const { colorScheme } = useContext(ThemeContext);  
  return <span>{colorScheme} logo</span>;  
};
```

Нужно передать созданный ранее контекст

Специальный хук для потребления контекста

Предвзятый совет:

- Старайтесь сопротивляться соблазну — используйте контексты как можно реже
- Причина: контексты ухудшают читаемость кода - вам будет очень сложно отслеживать, откуда берутся данные, и кто их меняет.

Где стоит использовать контекст

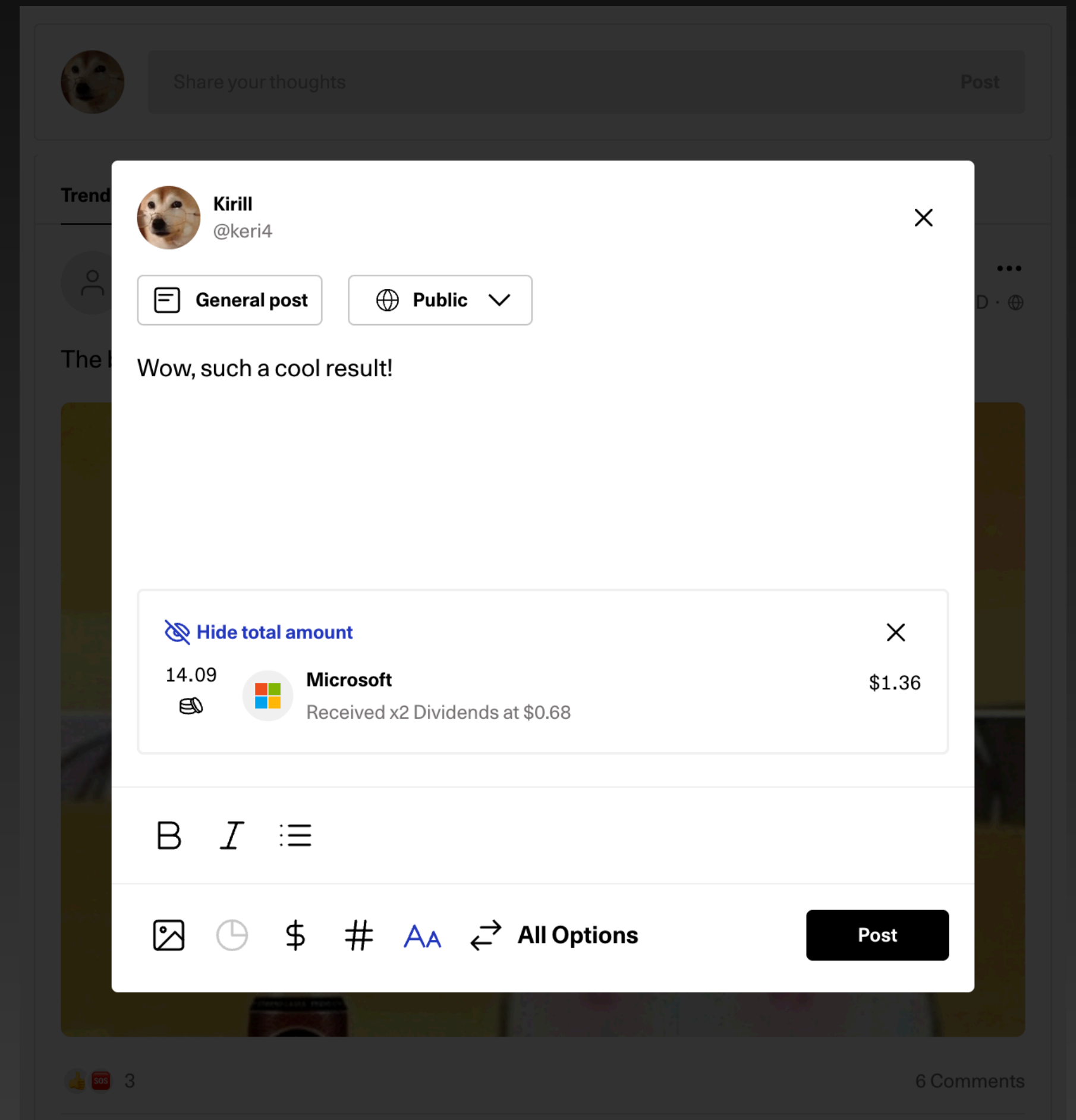
- Контекст полезен для глобальных данных, которые могут быть нужны всем элементам приложения
 - Язык интерфейса
 - Тема
 - Информация о пользователе
- Так же, очень редко бывает полезно использовать контекст для хранения локальных данных одной части приложения
 - Полезно только тогда, когда есть глубоко спрятанные элементы которые должны быть тесно связаны друг с другом
- Во всех остальных случаях, старайтесь обходиться пропсами.

Оптимизируем компоненты

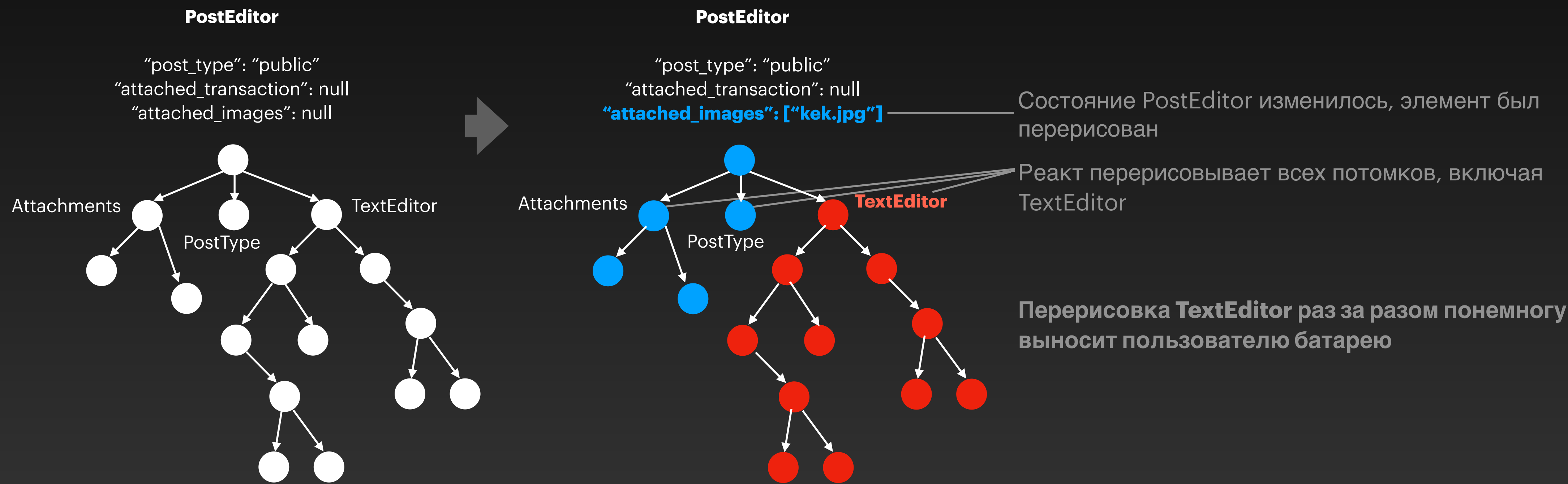
Memo API

Задача

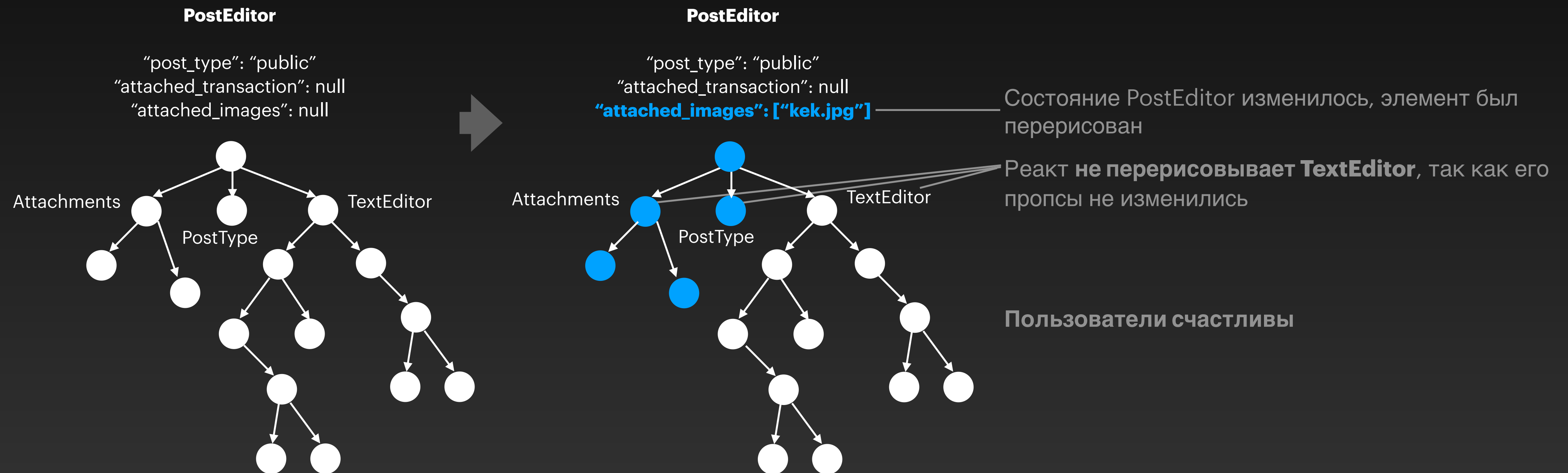
- У вас в приложении есть тяжелый и сложный компонент — текстовый редактор
 - Его перерисовка может занимать значительную часть ресурсов
- Ваши пользователи заметили, что когда они активно нажимают на элементы управления, батарея на их устройствах садится “за 2 минуты”
- Это происходит даже тогда, когда пользователи кликают на элементы управления, не относящиеся к редактору



Что происходит



Чего нужно добиться



React.memo()

Контролируем ре-рендер компонент

```
const HeavyComponent: FC = () => {  
  // этот компонент нужно рендерить как можно реже  
  return <div />;  
};  
  
// функция, для принятия решения о перерендере на основании пропсов  
// когда `return true`, компонент не будет перерисован  
function arePropsEqual<P>(prevProps: P, nextProps: P) {  
  // ваша логика сравнения пропсов  
  return true;  
}  
  
// новый компонент, который будет перерисовываться только на изменение пропсов  
export const MemoisedComponent = React.memo(HeavyComponent, arePropsEqual);
```