

Eloquent Code With React Hooks

Henrikas Kuzmickas



Wix Engineering Locations

Ukraine

Kiev
Dnipro

Israel

Tel-Aviv
Be'er Sheva

Lithuania

Vilnius





Note: this image is photoshopped

AGENDA

1. Hooks. What is it?
2. Hooks and State.
3. Hooks and GraphQL.

01

Hooks.
What is it?

Reusing stateful logic is hard.

```
▼ <Unknown>
  ▼ <t debug={false} errorMessage="">
    ▼ <o>
      ▼ <t>
        ▼ <t>
          ▼ <Router>
            ▼ <RouterContext>
              ▼ <Apollo(Connect(Apollo(n)))>
                ▼ <t fetchPolicy="network-only" errorPolicy="ignore" ssr={false} displayName="Apollo(Connect(Apollo(n)))" skip={false} warnUnhandledError={true}>
                  ▼ <Connect(Apollo(n)) authLoading={false} isAuthenticated={2539615}>
                    ▼ <Apollo(n) authLoading={false} isAuthenticated={2539615}>
                      ▼ <t errorPolicy="ignore" ssr={false} displayName="Apollo(n)" skip={false} warnUnhandledError={true}>
                        ▼ <n authLoading={false} isAuthenticated={2539615} userLoading={false}>
                          ▼ <Connect(Apollo(t)) authLoading={false} isAuthenticated={2539615} userLoading={false}>
                            ▶ <Apollo(t) authLoading={false} isAuthenticated={2539615} userLoading={false} isMobile={false} isOnline={true} lang="id" popUp={false} searchModalOpen={false} sessionId={2539615} xdevice="">...</Apollo(t)> == $r
                            </Connect(Apollo(t))>
                          </n>
                        </t>
                      </Apollo(n)>
                    </Connect(Apollo(n))>
                  </t>
                </Apollo(Connect(Apollo(n)))>
              </RouterContext>
            </Router>
          </t>
        </t>
      </o>
    </t>
  </Unknown>
```



Giant components suck.



```
1 class Example extends React.Component {
2   componentDidMount() {
3     this.subscribeToDataStore(this.props.thing.id);
4     this.fetchCommentsOrSomething(this.props.thing.id);
5     this.startTimers();
6   }
7
8   render() {
9     ...
10  }
11 }
```



```
1 class Example extends React.Component {
2   componentDidMount() {
3     this.subscribeToDataStore(this.props.thing.id);
4     this.fetchCommentsOrSomething(this.props.thing.id);
5     this.startTimers();
6   }
7
8   componentWillUnmount() {
9     this.unsubscribeFromDataStore();
10    this.cancelPendingRequests();
11    this.stopTimers();
12  }
13
14  render() {
15    ...
16  }
17 }
```

```
1 class Example extends React.Component {  
2   componentDidMount() {  
3     this.subscribeToDataStore(this.props.thing.id);  
4     this.fetchCommentsOrSomething(this.props.thing.id);  
5     this.startTimers();  
6   }  
7  
8   componentWillUnmount() {  
9     this.unsubscribeFromDataStore();  
10    this.cancelPendingRequests();  
11    this.stopTimers();  
12  }  
13  
14  componentDidUpdate() {  
15    ...🤔  
16  }  
17  
18  render() {  
19    ...  
20  }  
21 }
```

Classes suck.

Hard for Humans.

```
> class Cat {  
  miau() { return "miau" }  
}  
< undefined  


---



```
> const fluffykins = new Cat()
< undefined

```
> fluffykins.miau()  
< "miau"  


---



```
> typeof fluffykins
< "object"

```
> typeof Cat  
< "function"  


---



```
> |
```


```


```


```


```


```

Hard for Machines.

01

Hooks.
Ok, but how?

useState

```
1 import React, { useState } from 'react';
2
3 export const Counter = () => {
4   const [count, setCount] = useState(0);
5   const incrementCount = () => setCount(count + 1);
6
7   return (
8     <div>
9       <p>You clicked {count} times</p>
10      <button onClick={incrementCount}>Click Me</button>
11    </div>
12  )
13 }
```

```
1 import React from 'react';
2
3 class Counter extends React.Component {
4   constructor() {
5     this.state = { count: 0 };
6     this.incrementCount = this.incrementCount.bind(this);
7   }
8
9   incrementCount() {
10     this.setState({ count: this.state.count + 1 });
11   }
12
13   render() {
14     return (
15       <div>
16         <p>You clicked {this.state.count} times</p>
17         <button onClick={this.incrementCount}>Click Me</button>
18       </div>
19     );
20   }
21 }
22
23 export default Counter;
```

useEffect



```
1 import React, { Component, useState, useEffect } from 'react';
2
3 export const Counter = () => {
4   const [count, setCount] = useState(0);
5   const incrementCount = () => setCount(count + 1);
6
7   useEffect(() => {
8     document.title = `You clicked ${count} times`;
9   });
10
11  return (
12    <div>
13      <p>You clicked {count} times</p>
14      <button onClick={incrementCount}>Click me</button>
15    </div>
16  )
17 }
```

```
1 import React from 'react';
2
3 export class Counter extends React.Component {
4   constructor() {
5     this.state = { count: 0 };
6     this.incrementCount = this.incrementCount.bind(this);
7   }
8
9   incrementCount() {
10     this.setState({ count: this.state.count + 1 });
11   }
12
13   componentDidMount() {
14     document.title = `You clicked ${this.state.count} times`;
15   }
16
17   componentDidUpdate() {
18     document.title = `You clicked ${this.state.count} times`;
19   }
20
21   render() {
22     return (
23       <div>
24         <p>You clicked {this.state.count} times</p>
25         <button onClick={this.incrementCount}>Click Me</button>
26       </div>
27     );
28   }
29 }
```

before
hooks



after
hooks

doctors hate him



```
1 let subscription;
2
3 useEffect(() => {
4   subscription = props.source.subscribe();
5 });
```



```
1 let subscription;
2
3 useEffect(() => {
4   subscription = props.source.subscribe();
5 }, []);
```



```
1 let subscription;
2
3 useEffect(() => {
4   subscription = props.source.subscribe();
5 }, [props.source])
```



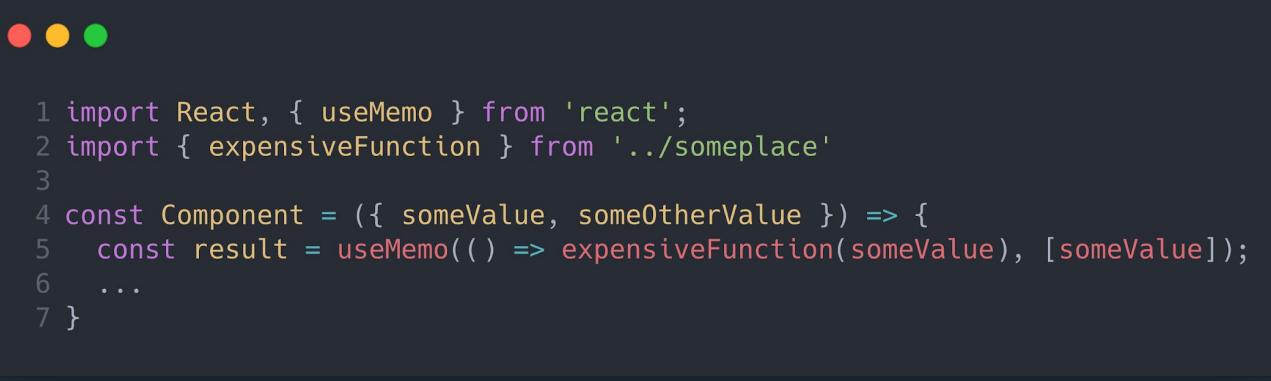
```
1 let subscription;
2
3 useEffect(() => {
4     subscription = props.source.subscribe();
5     return () => {
6         subscription.unsubscribe();
7     }
8 }, [props.source]);
```

useReducer

```
1 import React, { useReducer } from 'react';
2
3 const initialState = { count: 0 };
4
5 const reducer = (state, action) => {
6   switch (action.type) {
7     case 'increment':
8       return { count: state.count + 1 };
9     case 'decrement':
10       return { count: state.count - 1 };
11     default:
12       return state;
13   }
14 };
15
16 export const UseReducer = () => {
17   const [state, dispatch] = useReducer(reducer, initialState);
18
19   return (
20     <>
21       <p>Current Count: {state.count}</p>
22       <Button onClick={() => dispatch({ type: 'increment' })}>
23         Add One
24       </Button>
25       <Button onClick={() => dispatch({ type: 'decrement' })}>
26         Subtract One
27       </Button>
28     </>
29   );
30 };
```

useMemo

```
1 import React, { useMemo } from 'react';
2 import { expensiveFunction } from '../someplace'
3
4 const Component = ({ someValue, someOtherValue }) => {
5   const result = expensiveFunction(someValue);
6   ...
7 }
```



```
1 import React, { useMemo } from 'react';
2 import { expensiveFunction } from '../someplace'
3
4 const Component = ({ someValue, someOtherValue }) => {
5   const result = useMemo(() => expensiveFunction(someValue), [someValue]);
6   ...
7 }
```

```
1 import React, { useMemo } from 'react';
2 import { expensiveFunction } from '../someplace'
3
4 const Component = ({ someValue, someOtherValue }) => {
5   const result = useMemo(() => expensiveFunction(someValue), [someValue]);
6
7   return useMemo(() => (
8     <div>
9       <p>{someOtherValue}</p>
10      <p>{result}</p>
11    </div>
12  ), [result, someOtherValue])
13 }
```

```
1 import React, { useMemo } from 'react';
2 import { expensiveFunction } from '../someplace'
3
4 const Component = ({ someValue, someOtherValue }) => {
5   const result = useMemo(() => expensiveFunction(someValue), [someValue]);
6
7   return (
8     <div>
9       {useMemo(() => (<p>{someOtherValue}</p>), [someOtherValue])}
10      {useMemo(() => (<p>{result}</p>), [result])}
11     </div>
12   )
13 }
```

```
1 import React, { useState, useMemo } from 'react';
2 import faker from 'faker';
3
4 export const UseMemo = () => {
5   const [name, setName] = useState(faker.name.firstName());
6   const [color, setColor] = useState(faker.internet.color());
7
8   const setRandomColor = () => setColor(faker.internet.color());
9   const setRandomName = () => setName(faker.name.firstName());
10
11  return (
12    <>
13      <p style={{ color }}>Some Name: {name}</p>
14      {useMemo(
15        () => (
16          <p style={{ color }}>
17            Some Memoized Name: {name}
18          </p>
19        ),
20        [color]
21      )}
22      <button onClick={setRandomName}>
23        Set Random Name
24      </button>
25      <button onClick={setRandomColor}>
26        Set Random Color
27      </button>
28    </>
29  );
30};
```

Custom Hooks

```
1 import React, { useState, useEffect } from 'react';
2
3 function FriendStatus(props) {
4   const [isOnline, setIsOnline] = useState(null);
5
6   useEffect(() => {
7     function handleStatusChange(status) {
8       setIsOnline(status.isOnline);
9     }
10
11   ChatAPI.subscribeToFriendStatus(props.friend.id, handleStatusChange);
12   return () => {
13     ChatAPI.unsubscribeFromFriendStatus(props.friend.id, handleStatusChange);
14   };
15 });
16
17 if (isOnline === null) {
18   return 'Loading...';
19 }
20 return isOnline ? 'Online' : 'Offline';
21 }
```

```
1 import React, { useState, useEffect } from 'react';
2
3 function FriendListItem(props) {
4   const [isOnline, setIsOnline] = useState(null);
5
6   useEffect(() => {
7     function handleStatusChange(status) {
8       setIsOnline(status.isOnline);
9     }
10
11   ChatAPI.subscribeToFriendStatus(props.friend.id, handleStatusChange);
12   return () => {
13     ChatAPI.unsubscribeFromFriendStatus(props.friend.id, handleStatusChange);
14   };
15 });
16
17 return (
18   <li style={{ color: isOnline ? 'green' : 'black' }}>
19     {props.friend.name}
20   </li>
21 );
22 }
```

```
1 import React, { useState, useEffect } from 'react';
2
3 function useFriendStatus(friendID) {
4   const [isOnline, setIsOnline] = useState(null);
5
6   useEffect(() => {
7     function handleStatusChange(status) {
8       setIsOnline(status.isOnline);
9     }
10
11   ChatAPI.subscribeToFriendStatus(friendID, handleStatusChange);
12   return () => {
13     ChatAPI.unsubscribeFromFriendStatus(friendID, handleStatusChange);
14   };
15 });
16
17 return isOnline;
18 }
```



```
1 function FriendStatus(props) {  
2   const isOnline = useFriendStatus(props.friend.id);  
3  
4   if (isOnline === null) {  
5     return 'Loading...';  
6   }  
7   return isOnline ? 'Online' : 'Offline';  
8 }
```

```
1 function FriendListItem(props) {
2   const isOnline = useFriendStatus(props.friend.id);
3
4   return (
5     <li style={{ color: isOnline ? 'green' : 'black' }}>
6       {props.friend.name}
7     </li>
8   );
9 }
```

useDebounce

```
1 import React, { useState, useEffect } from 'react';
2
3 const useDebounce = (initVal, delay) => {
4   const [value, setValue] = useState(initVal);
5   const [debouncedValue, setDebouncedValue] = useState(initVal);
6
7   useEffect(() => {
8     const handler = setTimeout(() => setDebouncedValue(value), delay);
9     return () => clearTimeout(handler);
10   }, [value, delay]);
11
12   return { value, debouncedValue, setValue };
13 };
14
15 export const UseDebounce = () => {
16   const { value, debouncedValue, setValue } = useDebounce(' ', 500);
17   const onChange = e => setValue(e.target.value);
18
19   return (
20     <>
21       <input value={value} onChange={onChange} />
22       <p>Current debounced value: {debouncedValue}</p>
23     </>
24   );
25};
```

- These aren't all of the hooks that come by default in React
 - Even the most basic hooks can reduce boilerplate
- useMemo allows for precise optimization in function components
 - Custom hooks allow sharing stateful logic
 - Custom hooks allow for building new kinds of libraries
- Hooks have some rules and gotchas, be sure to read the docs

02

Hooks and
State.

Modals With Hooks

Our own Redux

- Multiple contexts allow for better separation of concerns
- You don't have to use 3rd party libs for global state
 - Typescript is easier with hooks

03

Hooks and
GraphQL.

Apollo without hooks

```
1 import React from 'react';
2 import { loader } from 'graphql.macro';
3 import { graphql } from 'react-apollo';
4
5 const USERS = loader('../apollo/queries/Users.graphql');
6
7 class SomeComponent extends React.Component {
8   renderUsers = () => {
9     return this.props.data.users.map(user => <div>{user.name}</div>);
10  };
11
12  render() {
13    if (this.props.data.loading) {
14      return <p>Loading...</p>;
15    }
16
17    return <div>{this.renderUsers()}</div>;
18  }
19}
20
21 export default graphql(USERS)(SomeComponent);
```

```
1 query Users {  
2   users {  
3     id  
4     name  
5     color {  
6       id  
7       value  
8     }  
9   }  
10 }
```

```
 1 import React from 'react';
 2 import { loader } from 'graphql.macro';
 3 import { graphQL } from 'react-apollo';
 4
 5 const USERS = loader('../apollo/queries/Users.graphql');
 6 const COLORS = loader('../apollo/queries/Colors.graphql');
 7
 8 class SomeComponent extends React.Component {
 9   renderUsers = () => {
10     return this.props.usersData.users.map(user => <div>{user.name}</div>);
11   };
12
13   renderColors = () => {
14     return this.props.colorsData.colors.map(color => <div>{color.name}</div>);
15   };
16
17   render() {
18     ...
19   }
20 }
21
22 export default graphQL(USERS, { name: 'usersData' })(  
23   graphQL(COLORS, { name: 'colorsData' }  
24 )(SomeComponent));
```

```
1 import React from 'react';
2 import { loader } from 'graphql.macro';
3 import { graphql, compose } from 'react-apollo';
4
5 const USERS = loader('../apollo/queries/Users.graphql');
6 const COLORS = loader('../apollo/queries/Colors.graphql');
7
8 class SomeComponent extends React.Component {
9   renderUsers = () => {
10     return this.props.usersData.users.map(user => <div>{user.name}</div>);
11   };
12
13   renderColors = () => {
14     return this.props.colorsData.colors.map(color => <div>{color.name}</div>);
15   };
16
17   render() {
18     ...
19   }
20 }
21
22 export default compose(
23   graphql(USERS, { name: 'usersData' }),
24   graphql(COLORS, { name: 'colorsData' })
25 )(SomeComponent);
```

```
1 import React from 'react';
2 import { loader } from 'graphql.macro';
3 import { Query } from 'react-apollo';
4
5 const USERS = loader('../apollo/queries/Users.graphql');
6
7 export class SomeComponent extends React.Component {
8     render() {
9         <Query query={USERS}>
10            {({ loading, data }) => {
11                if (loading) {
12                    return <p>Loading...</p>;
13                }
14
15                return (
16                    <ul>
17                        {data.users.map(user => (
18                            <li>{user.name}</li>
19                        ))}
20                    </ul>
21                );
22            }}
23        </Query>;
24    }
25 }
```

```
1 import React from 'react';
2 import { loader } from 'graphql.macro';
3 import { Query } from 'react-apollo';
4
5 const USERS = loader('../apollo/queries/Users.graphql');
6 const COLORS = loader('../apollo/queries/Colors.graphql');
7
8 export class SomeComponent extends React.Component {
9   render() {
10     <Query query={USERS}>
11       {({ loading: loadingUsers, data: userData }) => (
12         <Query query={COLORS}>
13           {({ loading: loadingColors, data: colorsData }) => {
14             if (loadingUsers || loadingColors) {
15               return <p>Loading...</p>;
16             }
17
18             return (
19               <div>
20                 {this.renderUsers(userData)}
21                 {this.renderColors(colorsData)}
22               </div>
23             );
24           }
25         </Query>
26       )}
27     </Query>;
28   }
29 }
```

```
● ● ●
```

```
1 import React from 'react';
2 import { Loader } from 'graphql.macro';
3 import { Query } from 'react-apollo';
4 import { adopt } from 'react-adopt';
5
6 const USERS = loader('../apollo/queries/Users.graphql');
7 const COLORS = loader('../apollo/queries/Colors.graphql');
8
9 export class SomeComponent extends React.Component {
10   render() {
11     const Composed = adopt({
12       userData: ({ render }) => <Query query={USERS}>{render}</Query>,
13       colorsData: ({ render }) => <Query query={COLORS}>{render}</Query>
14     });
15
16     return (
17       <Composed>
18         {({ userData, colorsData }) => {
19           if (userData.loading || colorsData.loading) {
20             return <p>Loading...</p>;
21           }
22
23           return (
24             <div>
25               {this.renderUsers(userData)}
26               {this.renderColors(colorsData)}
27             </div>
28           );
29         }}
30       </Composed>
31     );
32   }
33 }
```

```
● ● ●

1 import React from 'react';
2 import { loader } from 'graphql.macro';
3 import { graphql, compose, GraphqlQueryControls } from 'react-apollo';
4 import { Users, Colors } from '../../../../../generated/schema';
5
6 const USERS = loader('../apollo/queries/Users.graphql');
7 const COLORS = loader('../apollo/queries/Colors.graphql');
8
9 interface Props {
10   userData: GraphqlQueryControls & Users;
11   colorsData: GraphqlQueryControls & Colors;
12 }
13
14 class SomeComponent extends React.Component<Props> {
15   renderUsers = () => {
16     return this.props.userData.users.map(user => <div>{user.name}</div>);
17   };
18
19   renderColors = () => {
20     return this.props.colorsData.colors.map(color => <div>{color.value}</div>);
21   };
22
23   render() {
24     if (this.props.userData.loading || this.props.colorsData.loading) {
25       return <p>Loading...</p>;
26     }
27
28     return (
29       <div>
30         {this.renderUsers()}
31         {this.renderColors()}
32       </div>
33     );
34   }
35 }
36
37 export default compose(
38   graphql(USERS, { name: 'userData' }),
39   graphql(COLORS, { name: 'colorsData' })
40 )(SomeComponent);
```

```
1 import React, { Component } from 'react';
2 import { Container, Card, CardContent, Typography, Button } from 'material-ui/core';
3 import { Container, Box, Grid, Paper, ButtonBase, Avatar, IconButton } from 'material-ui';
4 import { User, Colors, CreateUser, CreateUserVariables } from '../generated/schema';
5
6 import { USERS, loader, apolloClient } from './utils';
7 import { CREATE_USER, CREATE_COLOR } from './queries';
8
9 interface Props {
10   user: User;
11   colors: Colors;
12   onCreateUser: () => void;
13   onCreateColor: () => void;
14 }
15
16 class UserListComponent extends Component<Props> {
17   state = {
18     users: [],
19     loading: false,
20   };
21
22   componentDidMount() {
23     if (!this.props.user.id) {
24       this.props.onCreateUser();
25     }
26   }
27
28   return this.props.user.id ? <> <div style={{ background-color: user.color.value }}>
29     <UserCard user={user} colors={colors} onCreateUser={onCreateUser} onCreateColor={onCreateColor} />
30   </div> : <> <div>
31     <UserList users={users} onCreateUser={onCreateUser} onCreateColor={onCreateColor} />
32   </div>;
33 }
34
35 class UserCard extends Component<{ user: User, colors: Colors, onCreateUser: () => void, onCreateColor: () => void }> {
36   render() {
37     const color = colors.colors[Math.floor(Math.random()) * colors.length];
38     const name = user.name || `Random-${user.id}`;
39
40     return <div style={{ border: '1px solid #ccc', padding: '10px' }}>
41       <div>
42         <Avatar style={{ color: color }}>{name}</Avatar>
43         <UserDetails user={user} colors={colors} />
44       </div>
45       <div>
46         <UserActions user={user} colors={colors} onCreateUser={onCreateUser} onCreateColor={onCreateColor} />
47       </div>
48     </div>;
49   }
50 }
51
52 class UserList extends Component<{ users: User[], onCreateUser: () => void }> {
53   render() {
54     const data = proxy.readQuery<User>({ query: USERS });
55
56     if (data) {
57       data.users.push(res.data.createUser);
58       proxy.writeQuery<User>({ query: USERS, data });
59     }
60
61     return <div>
62       <Table users={users} onCreateUser={onCreateUser} />
63     </div>;
64   }
65 }
66
67 class CreateUser extends Component<{ onCreateUser: () => void }> {
68   state = {
69     value: '',
70   };
71
72   handleChange = (e) => {
73     this.setState({ value: e.target.value });
74   };
75
76   handleSubmit = (e) => {
77     e.preventDefault();
78     const user = {
79       id: 'random-' + Date.now(),
80       name: this.state.value,
81       color: colors.colors[Math.floor(Math.random()) * colors.length],
82     };
83
84     proxy.writeQuery<User>({ query: CREATE_USER, variables: { user } });
85
86     this.props.onCreateUser();
87   }
88
89   render() {
90     return <div>
91       <Form>
92         <Input type="text" value={value} onChange={this.handleChange} />
93         <Button type="submit" onClick={this.handleSubmit}>Create New User</Button>
94       </Form>
95     </div>;
96   }
97 }
98
99 class CreateColor extends Component<{ onCreateColor: () => void }> {
100   state = {
101     value: '',
102   };
103
104   handleChange = (e) => {
105     this.setState({ value: e.target.value });
106   };
107
108   handleSubmit = (e) => {
109     e.preventDefault();
110     const color = {
111       id: 'random-' + Date.now(),
112       value: this.state.value,
113     };
114
115     proxy.writeQuery<Color>({ query: CREATE_COLOR, variables: { color } });
116
117     this.props.onCreateColor();
118   }
119
120   render() {
121     return <div>
122       <Form>
123         <Input type="text" value={value} onChange={this.handleChange} />
124         <Button type="submit" onClick={this.handleSubmit}>Create New Color</Button>
125       </Form>
126     </div>;
127   }
128 }
129
130 export const userList = composed(
131   graphql(USERS, { name: 'createUser' }),
132   graphql(USERS, { name: 'createColor' }),
133   graphql(CREATE_USER, { name: 'createUser' }),
134   graphql(CREATE_COLOR, { name: 'createColor' })
135 )(UserListComponent);
```

With Hooks

```
1 import React from 'react';
2 import { loader } from 'graphql.macro';
3 import { useQuery } from '@apollo/react-hooks';
4
5 const USERS = loader('../apollo/queries/Users.graphql');
6
7 export const SomeComponent = () => {
8   const usersQuery = useQuery(USERS);
9
10  if (usersQuery.loading || !usersQuery.data) {
11    return <p>Loading...</p>;
12  }
13
14  return (
15    <div>
16      {usersQuery.data.users.map(user => (
17        <div>{user.name}</div>
18      ))}
19    </div>
20  );
21};
```

```
1 import React from 'react';
2 import { loader } from 'graphql.macro';
3 import { useQuery } from '@apollo/react-hooks';
4 import { Users } from '../../generated/schema';
5
6 const USERS = loader('../apollo/queries/Users.graphql');
7
8 export const SomeComponent = () => {
9   const usersQuery = useQuery<Users>(USERS);
10
11   if (usersQuery.loading || !usersQuery.data) {
12     return <p>Loading...</p>;
13   }
14
15   return (
16     <div>
17       {usersQuery.data.users.map(user => (
18         <div>{user.name}</div>
19       ))}
20     </div>
21   );
22};
```

```
1 import React from 'react';
2 import { loader } from 'graphql.macro';
3 import { useQuery } from '@apollo/react-hooks';
4 import { Users, Colors } from '../../generated/schema';
5
6 const USERS = loader('../apollo/queries/Users.graphql');
7 const COLORS = loader('../apollo/queries/Colors.graphql');
8
9 export const SomeComponent = () => {
10   const usersQuery = useQuery<Users>(USERS);
11   const colorsQuery = useQuery<Colors>(COLORS);
12
13   if (!colorsQuery.data || !usersQuery.data) {
14     return <p>Loading...</p>;
15   }
16
17   return (
18     <div>
19       {usersQuery.data.users.map(user => (
20         <div>{user.name}</div>
21       ))}
22       {colorsQuery.data.colors.map(color => (
23         <div>{color.value}</div>
24       ))}
25     </div>
26   );
27 };
```

```
● ● ●

1 import React from 'react';
2 import { loader } from 'graphql.macro';
3 import { useMutation } from '@apollo/react-hooks';
4
5 const CREATE_USER = loader('../apollo/queries/CreateUser.graphql');
6
7 export const SomeComponent = () => {
8   const [createUserMutation] = useMutation(CREATE_USER);
9
10  const createUser = () => {
11    createUserMutation({
12      variables: {
13        name: 'John'
14      }
15    });
16  };
17
18  return (
19    <div>
20      <button onClick={createUser}>Create User</button>
21    </div>
22  );
23}
```

```
1 import React from 'react';
2 import { loader } from 'graphql.macro';
3 import { useMutation } from '@apollo/react-hooks';
4 import { CreateUser, CreateUserVariables } from '../../generated/schema';
5
6 const CREATE_USER = loader('../apollo/queries/CreateUser.graphql');
7
8 export const SomeComponent = () => {
9   const [createUserMutation] = useMutation<CreateUser, CreateUserVariables>(CREATE_USER);
10
11   const createUser = () => {
12     createUserMutation({
13       variables: {
14         name: 'John'
15       }
16     );
17   };
18
19   return (
20     <div>
21       <button onClick={createUser}>Create User</button>
22     </div>
23   );
24 }
```

```
1 import React from 'react';
2 import { loader } from 'graphql.macro';
3 import { useMutation } from '@apollo/react-hooks';
4 import { CreateUser, CreateUserVariables } from '../../../../../generated/schema';
5
6 const CREATE_USER = loader('../apollo/queries/CreateUser.graphql');
7
8 export const SomeComponent = () => {
9   const [createUserMutation] = useMutation<CreateUser, CreateUserVariables>(CREATE_USER);
10
11   const createUser = () => {
12     createUserMutation({
13       variables: {
14         name: 'John'
15       },
16       optimisticResponse: {
17         createUser: {
18           __typename: 'User',
19           id: 'random-id',
20           name: 'John',
21           color: {
22             __typename: 'Color',
23             id: 'random-id',
24             value: 'red'
25           }
26         }
27       }
28     });
29   };
30
31   return (
32     <div>
33       <button onClick={createUser}>Create User</button>
34     </div>
35   );
36};
```

```
1 import React from 'react';
2 import { loader } from 'graphql.macro';
3 import { useMutation } from '@apollo/react-hooks';
4 import { CreateUser, CreateUserVariables, Users } from '../../../../../generated/schema';
5
6 const USERS = loader('../apollo/queries/Users.graphql');
7 const CREATE_USER = loader('../apollo/queries/CreateUser.graphql');
8
9 export const SomeComponent = () => {
10   const [createUserMutation] = useMutation<CreateUser, CreateUserVariables>(CREATE_USER);
11
12   const createUser = () => {
13     createUserMutation({
14       variables: {
15         name: 'John'
16       },
17       optimisticResponse: {
18         createUser: {
19           __typename: 'User',
20           id: 'random-id',
21           name: 'John',
22           color: {
23             __typename: 'Color',
24             id: 'random-id',
25             value: 'red'
26           }
27         }
28       },
29       update: (proxy, res) => {
30         if (!res.data || !res.data.createUser) {
31           return;
32         }
33
34         const data = proxy.readQuery<Users>({ query: USERS });
35
36         if (data) {
37           data.users.push(res.data.createUser);
38           proxy.writeQuery({ query: USERS, data });
39         }
40       }
41     });
42   };
43
44   return (
45     <div>
46       <button onClick={createUser}>Create User</button>
47     </div>
48   );
49};
```



```
1 import React from 'react';
2 import { useMutations } from '../hooks';
3
4 export const SomeComponent = () => {
5   const { createUser } = useMutations();
6
7   return (
8     <div>
9       <button onClick={createUser}>Create User</button>
10    </div>
11  );
12};
```

```
1 import React from 'react';
2 import { useUsers, useColors, useMutations } from '../hooks';
3
4 export const UserList = () => {
5   const [users] = useUsers();
6   const [colors] = useColors();
7   const { createUser, createColor } = useMutations();
8
9   const renderUsers = () => {
10     return users ? users.map(user => <p>{user.name}</p>) : 'Loading...';
11   };
12
13   const renderColors = () => {
14     return colors ? colors.map(color => <p>{color.value}</p>) : 'Loading...';
15   };
16
17   return (
18     <div>
19       {renderUsers()}
20       {renderColors()}
21       <button onClick={createUser} size="small" color="primary">
22         Create New User
23       </button>
24       <button onClick={createColor} size="small" color="primary">
25         Create New Color
26       </button>
27     </div>
28   );
29 }
```

```
1 import React, { Component } from 'react';
2 import { Container, Card, CardContent, Typography, Button } from 'material-ui/core';
3 import { Container, Box, Grid, Paper, ButtonBase, Avatar, IconButton } from 'material-ui';
4 import { User, Colors, CreateUser, CreateUserVariables } from '../generated/schema';
5
6 import { USERS, loader, apolloClient } from './utils';
7 import { CREATE_USER, CREATE_COLOR } from './mutations';
8 import { COLOR } from './queries';
9
10 interface Props {
11   user: User;
12   colors: Colors;
13   onCreateUser: () => void;
14   onCreateColor: () => void;
15 }
16
17 class UserListComponent extends Component<Props> {
18   state = {
19     users: [],
20     loading: false,
21   };
22
23   componentDidMount() {
24     if (!this.props.user.id) {
25       this.fetchUser();
26     }
27   }
28
29   render() {
30     const users = this.state.users.map((user) => {
31       return (
32         

33           
34           

35             

{user.name}


36             

{user.color}


37


38


39       );
40     });
41
42     return (
43       

44         

## User List


45         

46           &plus;
47           &plus;
48

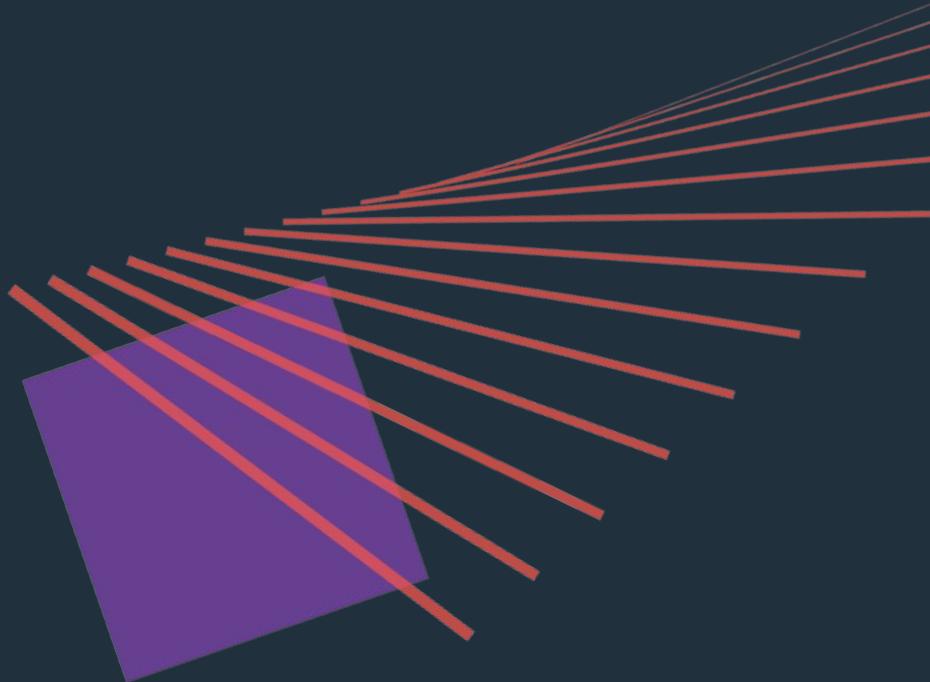

49         

50           {users}
51


52


53     );
54   }
55
56   fetchUser = async () => {
57     const res = await apolloClient.query({
58       query: USER,
59       variables: { id: this.props.user.id },
60     });
61
62     if (res.data) {
63       this.setState({ user: res.data.user });
64     }
65   }
66
67   onCreateUser = async () => {
68     const res = await apolloClient.mutate({
69       mutation: CREATE_USER,
70       variables: { name: 'Faker' },
71     });
72
73     if (res.data) {
74       this.setState({ users: [...this.state.users, res.data.createUser] });
75     }
76   }
77
78   onCreateColor = async () => {
79     const res = await apolloClient.mutate({
80       mutation: CREATE_COLOR,
81       variables: { color: '#ff0000' },
82     });
83
84     if (res.data) {
85       this.setState({ colors: [...this.state.colors, res.data.createColor] });
86     }
87   }
88
89   render() {
90     return (
91       <div style={{ width: '100%' }>
92         <Container>
93           <Grid container style={{ padding: 10 }}>
94             <Grid item style={{ flex: 1 }}>
95               <Card style={{ border: 1px solid #ccc, padding: 10, border-radius: 5px }}>
96                 <CardContent>
97                   <Typography variant="h2" color="textSecondary">User List</Typography>
98                   <Table border="1" style={{ width: '100%', border-collapse: 'collapse' }}>
99                     <thead>
100                       <tr>
101                         <th style={{ padding: 5px, text-align: 'left' }}>Name</th>
102                         <th style={{ padding: 5px, text-align: 'left' }}>Color</th>
103                       </tr>
104                     </thead>
105                     <tbody>
106                       <tr>
107                         <td style={{ padding: 5px, vertical-align: 'top' }}>
108                           <img alt="User icon" style={{ width: 24 }}/>
109                           <div style={{ margin-top: 5px }}>
110                             <p>Faker</p>
111                             <p>#ff0000</p>
112                           </div>
113                         </td>
114                         <td style={{ padding: 5px, text-align: 'right' }}>
115                           <button style={{ border: 1px solid #ccc, padding: 2px 5px, border-radius: 3px }}>&minus;</button>
116                           <button style={{ border: 1px solid #ccc, padding: 2px 5px, border-radius: 3px }}>&plus;</button>
117                         </td>
118                       </tr>
119                     </tbody>
120                   </Table>
121                 </CardContent>
122               </Card>
123             </Grid>
124           </Grid>
125         </Container>
126       </div>
127     );
128   }
129 }
130
131 export const userList = compose(
132   graphql(USERS, { name: 'createUser' }),
133   graphql(COLORS, { name: 'createColor' }),
134   graphql(CREATE_USER, { name: 'createUser' }),
135   graphql(CREATE_COLOR, { name: 'createColor' })
136 )(UserListComponent);
```

Thank You



Q&A

