WiX Engineering

Eloquent Code With React Hooks

Henrikas Kuzmickas

henrikask@wix.com twitter@henry_kuzmick github.com/henrykuzmick

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.

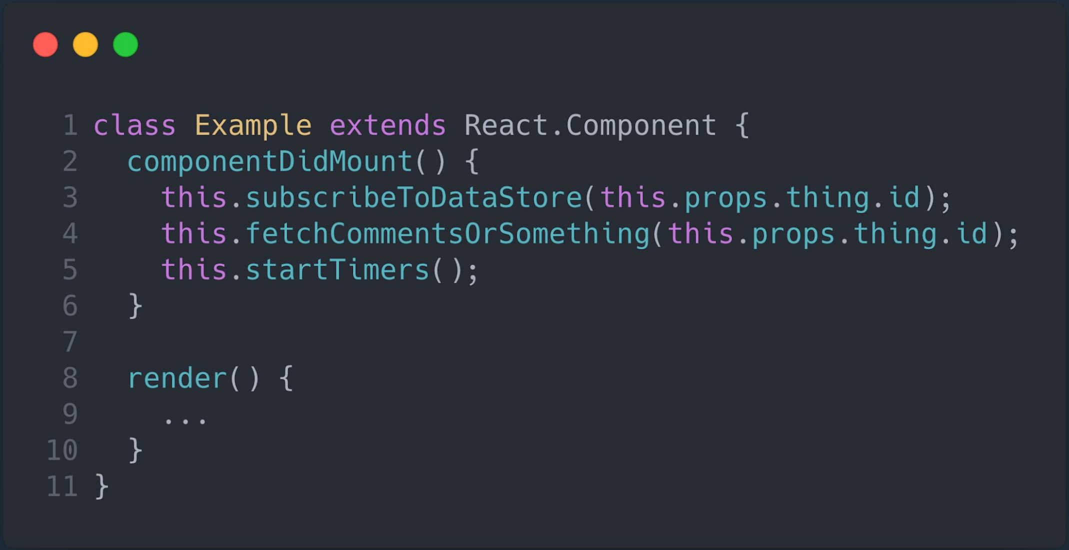
01

Hooks.
What is it?

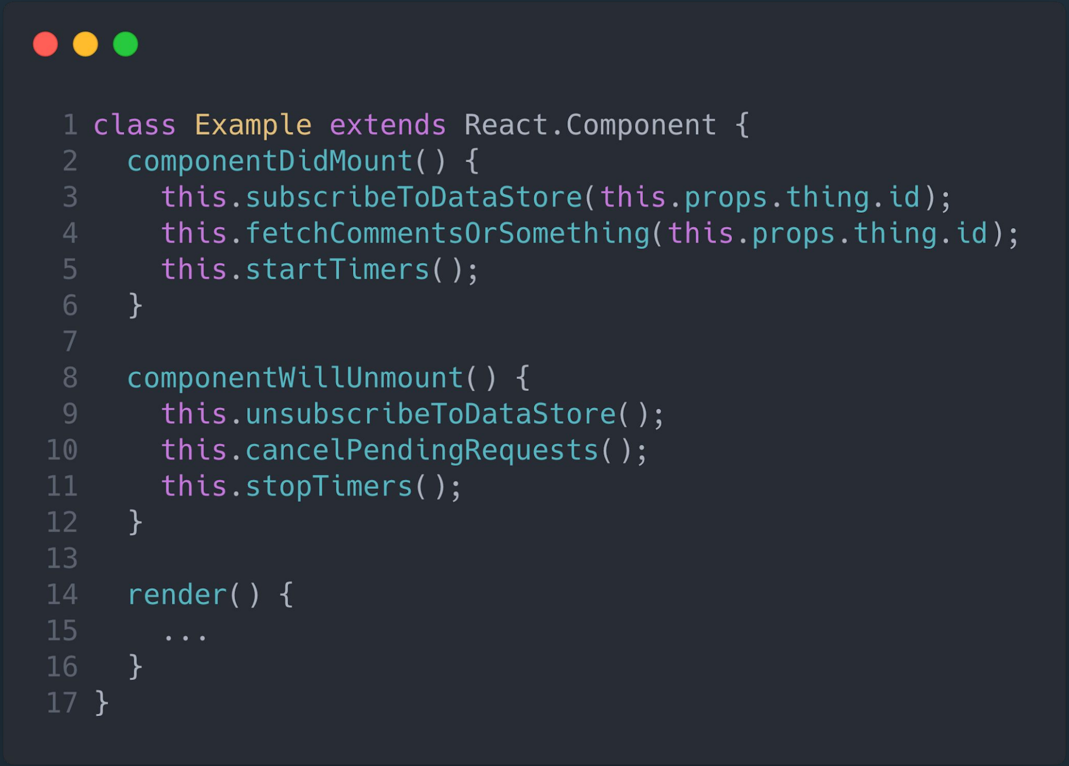
Reusing stateful logic is hard.

```
▼ <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>
```

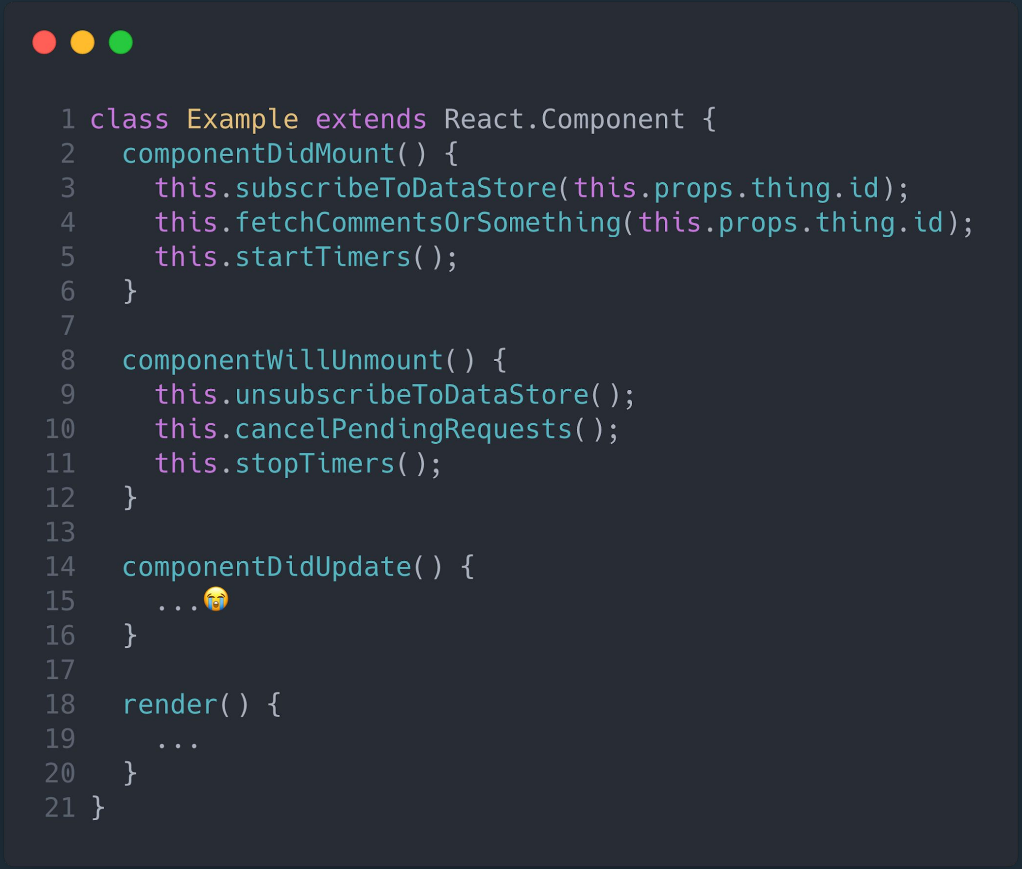
Giant components suck.



```
class Example extends React.Component {
  componentDidMount() {
    this.subscribeToDataStore(this.props.thing.id);
    this.fetchCommentsOrSomething(this.props.thing.id);
    this.startTimers();
  }

  render() {
    ...
  }
}
```

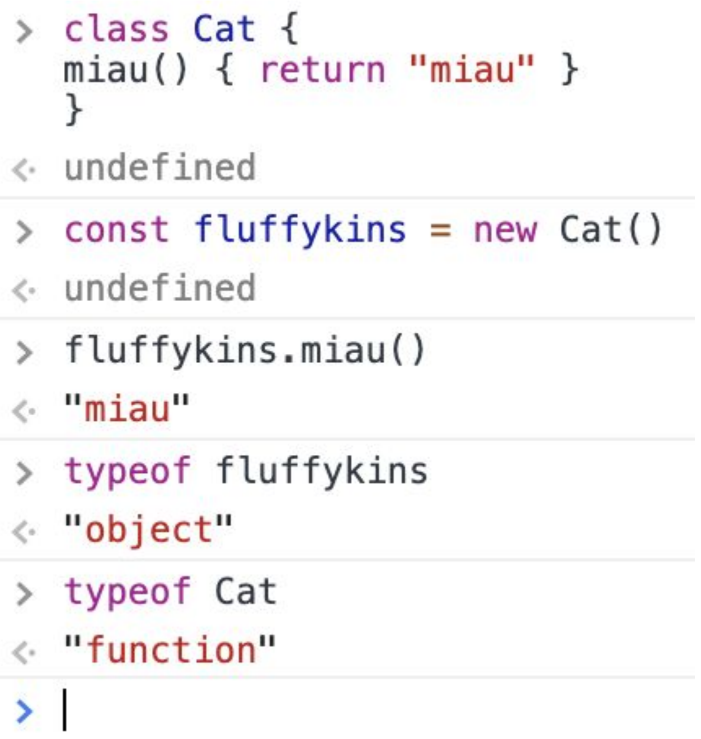


```
class Example extends React.Component {
  componentDidMount() {
    this.subscribeToDataStore(this.props.thing.id);
    this.fetchCommentsOrSomething(this.props.thing.id);
    this.startTimers();
  }

  componentWillUnmount() {
    this.unsubscribeToDataStore();
    this.cancelPendingRequests();
    this.stopTimers();
  }

  render() {
    ...
  }
}
```


```
class Example extends React.Component {
  componentDidMount() {
    this.subscribeToDataStore(this.props.thing.id);
    this.fetchCommentsOrSomething(this.props.thing.id);
    this.startTimers();
  }

  componentWillUnmount() {
    this.unsubscribeToDataStore();
    this.cancelPendingRequests();
    this.stopTimers();
  }

  componentDidUpdate() {
    ...😭
  }

  render() {
    ...
  }
}
```

Classes suck.

Hard for Humans.



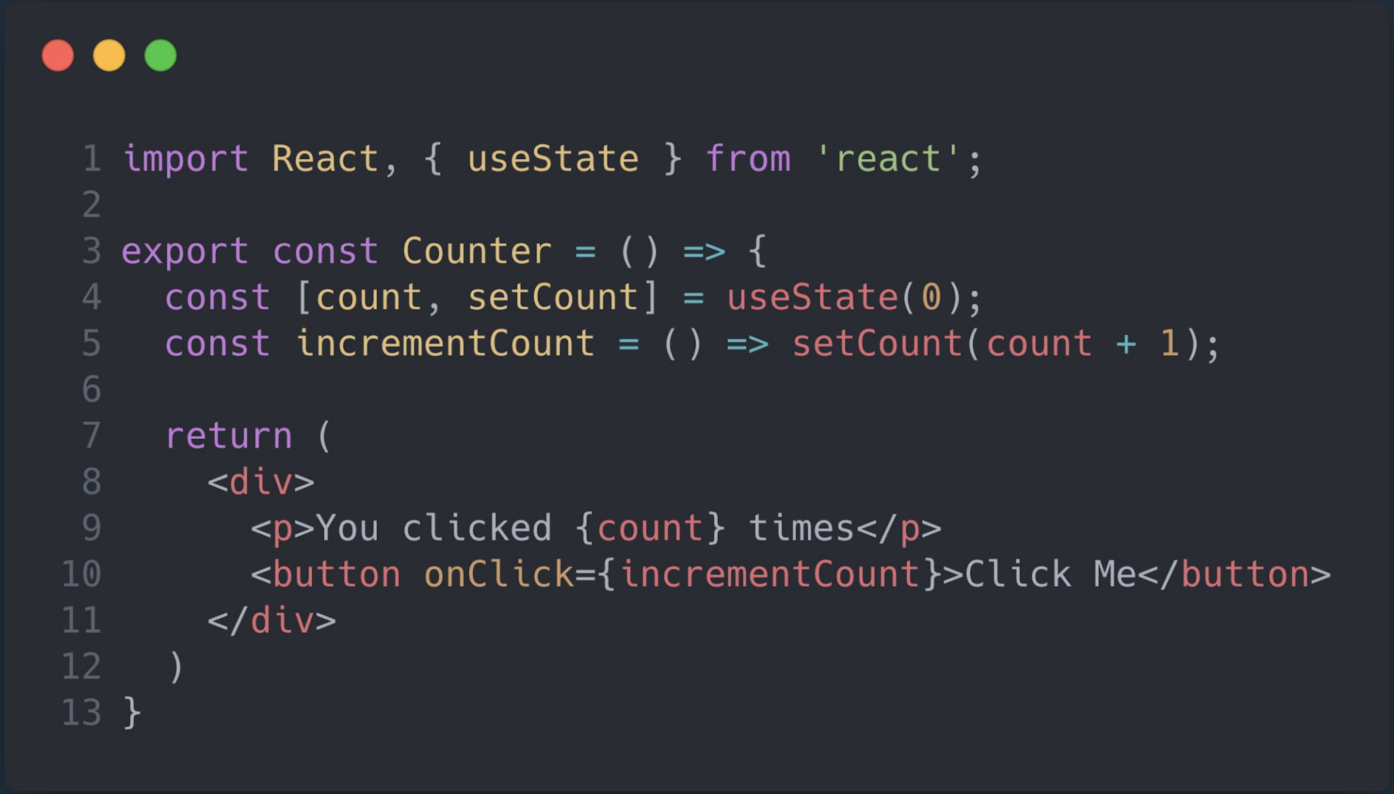
```
> class Cat {
  miau() { return "miau" }
  }
<· undefined
> const fluffykins = new Cat()
<· undefined
> fluffykins.miau()
<· "miau"
> typeof fluffykins
<· "object"
> typeof Cat
<· "function"
> |
```

Hard for Machines.

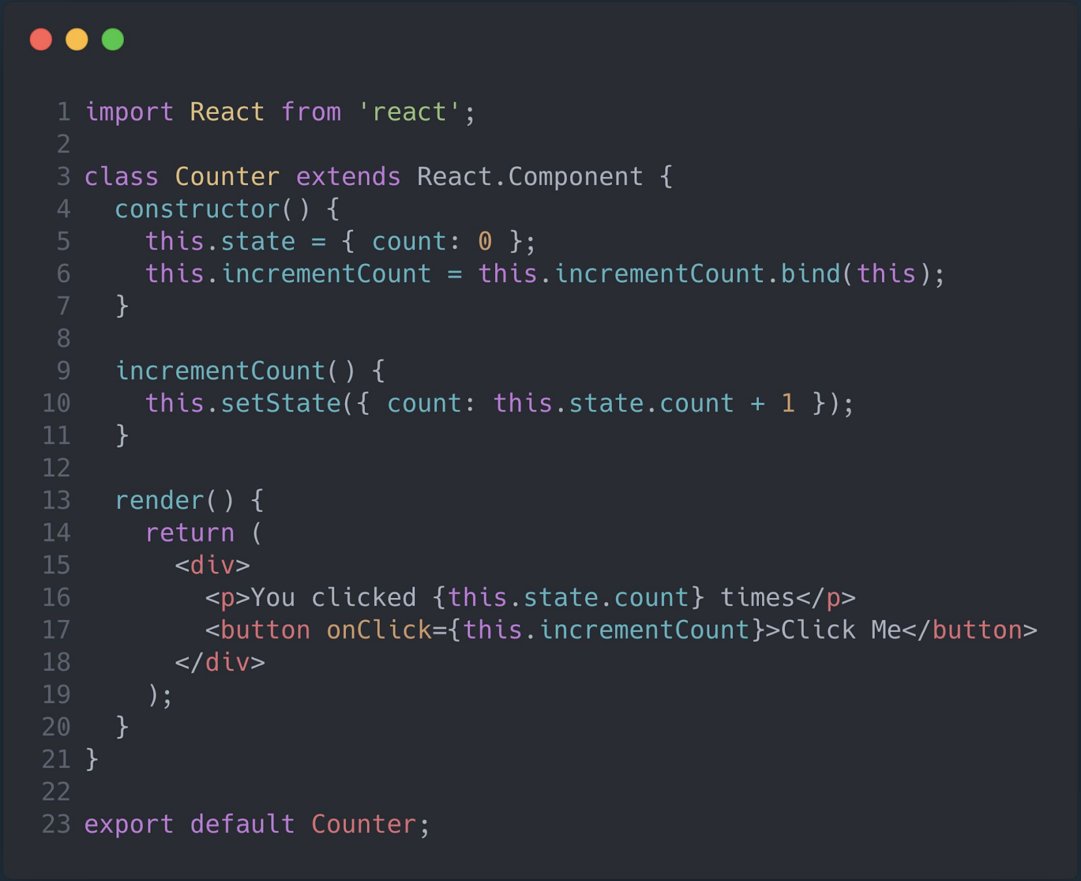
01

Hooks.
Ok, but how?

useState

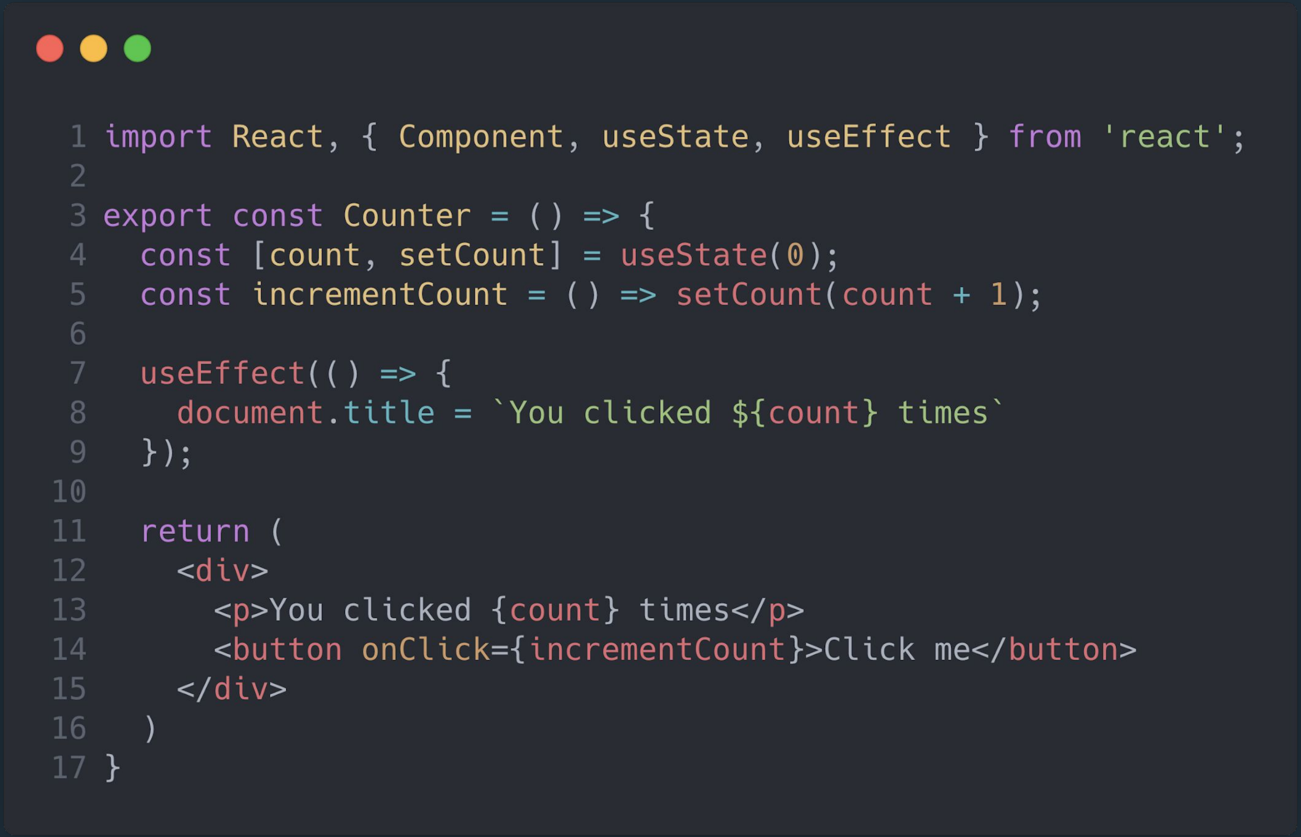


```jsx
import React, { useState } from 'react';

export const Counter = () => {
  const [count, setCount] = useState(0);
  const incrementCount = () => setCount(count + 1);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={incrementCount}>Click Me</button>
    </div>
  )
}
```

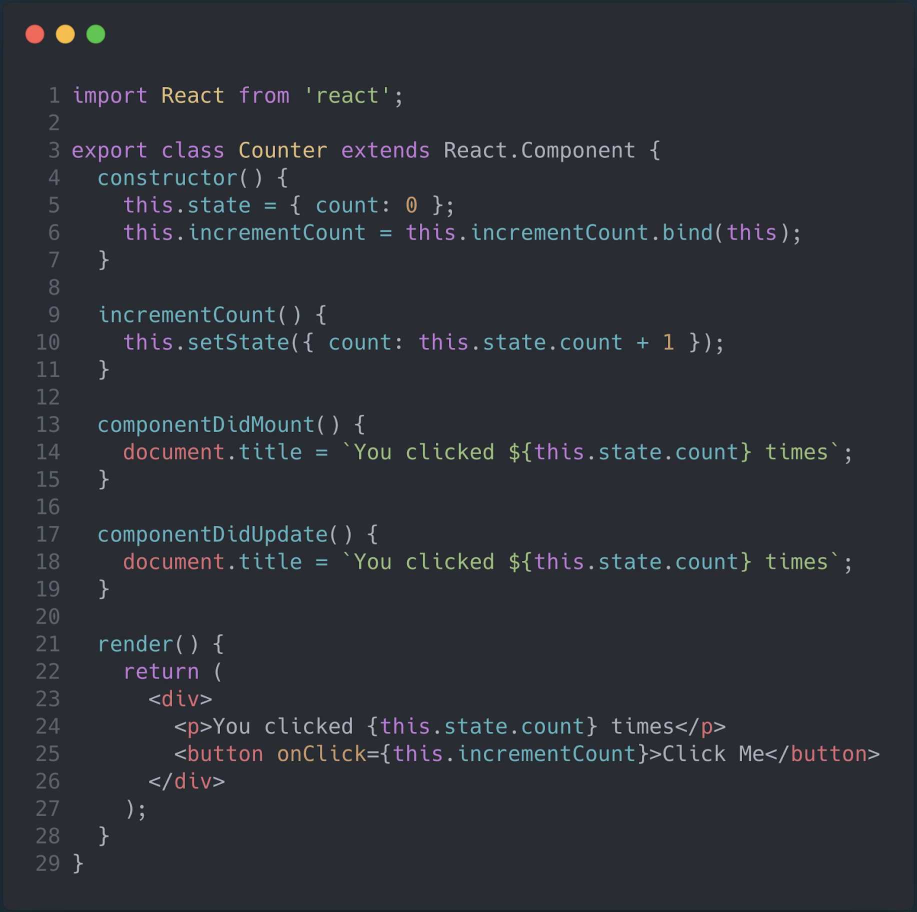


```jsx
import React from 'react';

class Counter extends React.Component {
  constructor() {
    this.state = { count: 0 };
    this.incrementCount = this.incrementCount.bind(this);
  }

  incrementCount() {
    this.setState({ count: this.state.count + 1 });
  }

  render() {
    return (
      <div>
        <p>You clicked {this.state.count} times</p>
        <button onClick={this.incrementCount}>Click Me</button>
      </div>
    );
  }
}

export default Counter;
```

useEffect



```
import React, { Component, useState, useEffect } from 'react';

export const Counter = () => {
  const [count, setCount] = useState(0);
  const incrementCount = () => setCount(count + 1);

  useEffect(() => {
    document.title = `You clicked ${count} times`
  });

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={incrementCount}>Click me</button>
    </div>
  )
}
```

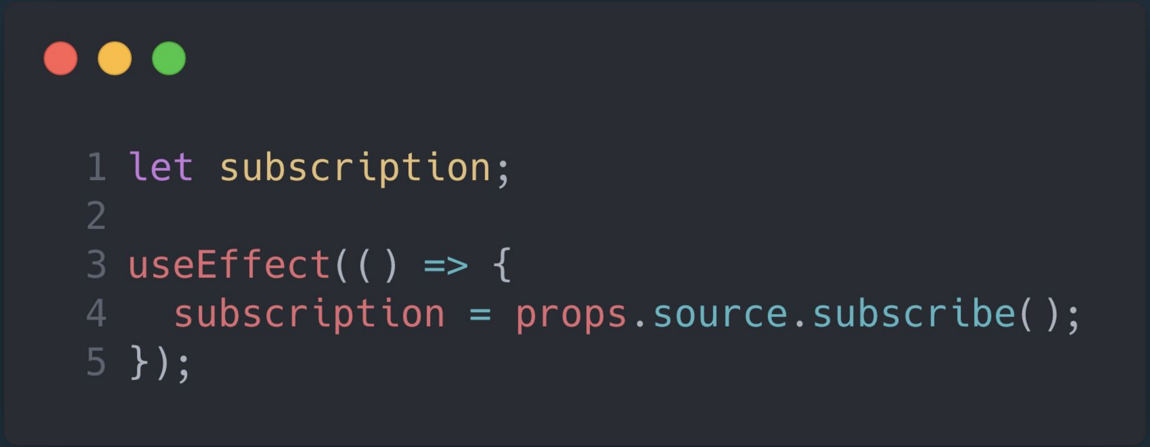


```jsx
import React from 'react';

export class Counter extends React.Component {
  constructor() {
    this.state = { count: 0 };
    this.incrementCount = this.incrementCount.bind(this);
  }

  incrementCount() {
    this.setState({ count: this.state.count + 1 });
  }

  componentDidMount() {
    document.title = `You clicked ${this.state.count} times`;
  }

  componentDidUpdate() {
    document.title = `You clicked ${this.state.count} times`;
  }

  render() {
    return (
      <div>
        <p>You clicked {this.state.count} times</p>
        <button onClick={this.incrementCount}>Click Me</button>
      </div>
    );
  }
}
```

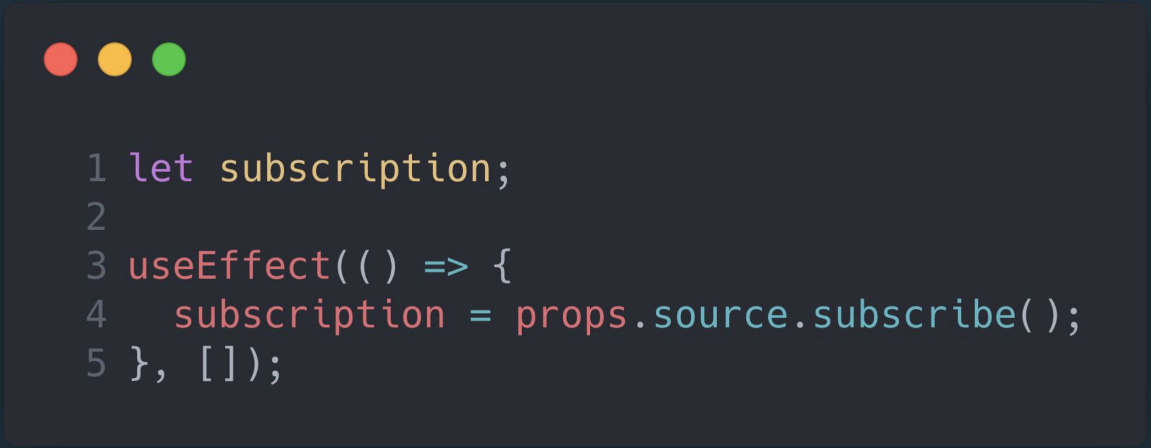
before hooks

after hooks

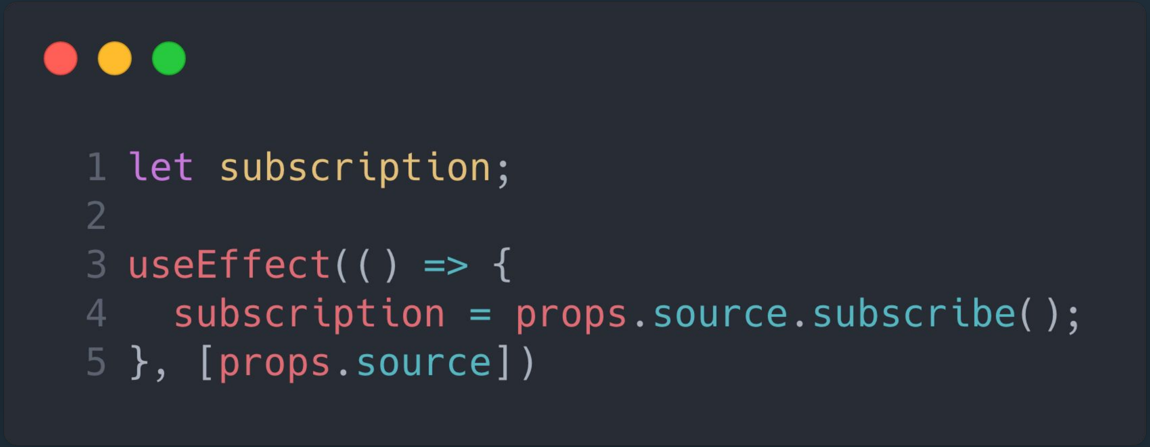
doctors hate him



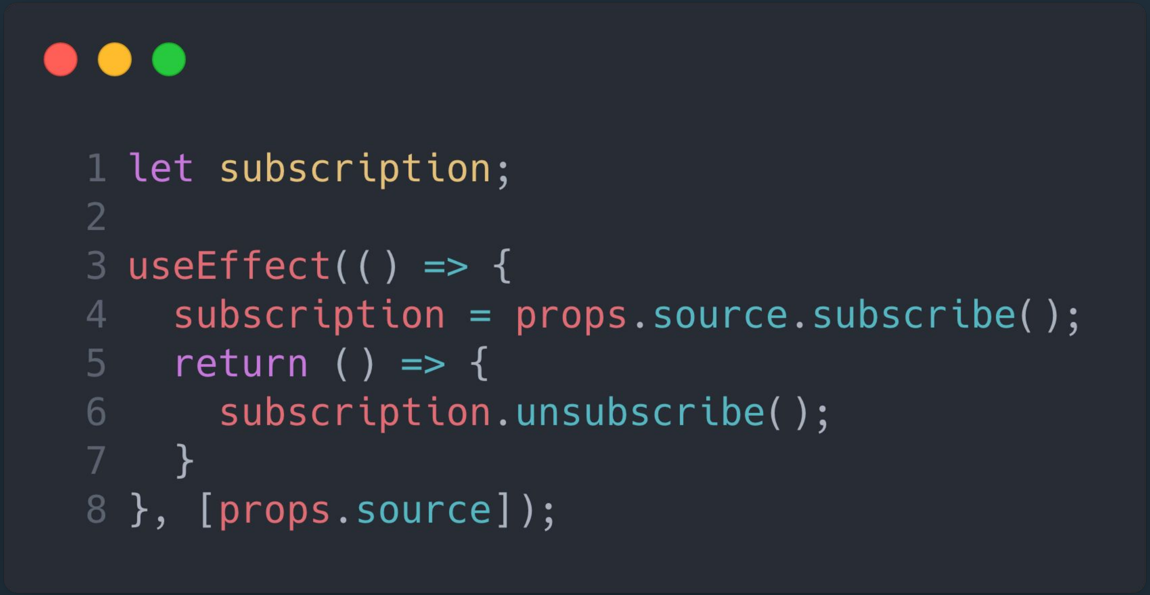
```javascript
let subscription;

useEffect(() => {
  subscription = props.source.subscribe();
});
```



```
let subscription;

useEffect(() => {
  subscription = props.source.subscribe();
}, []);
```

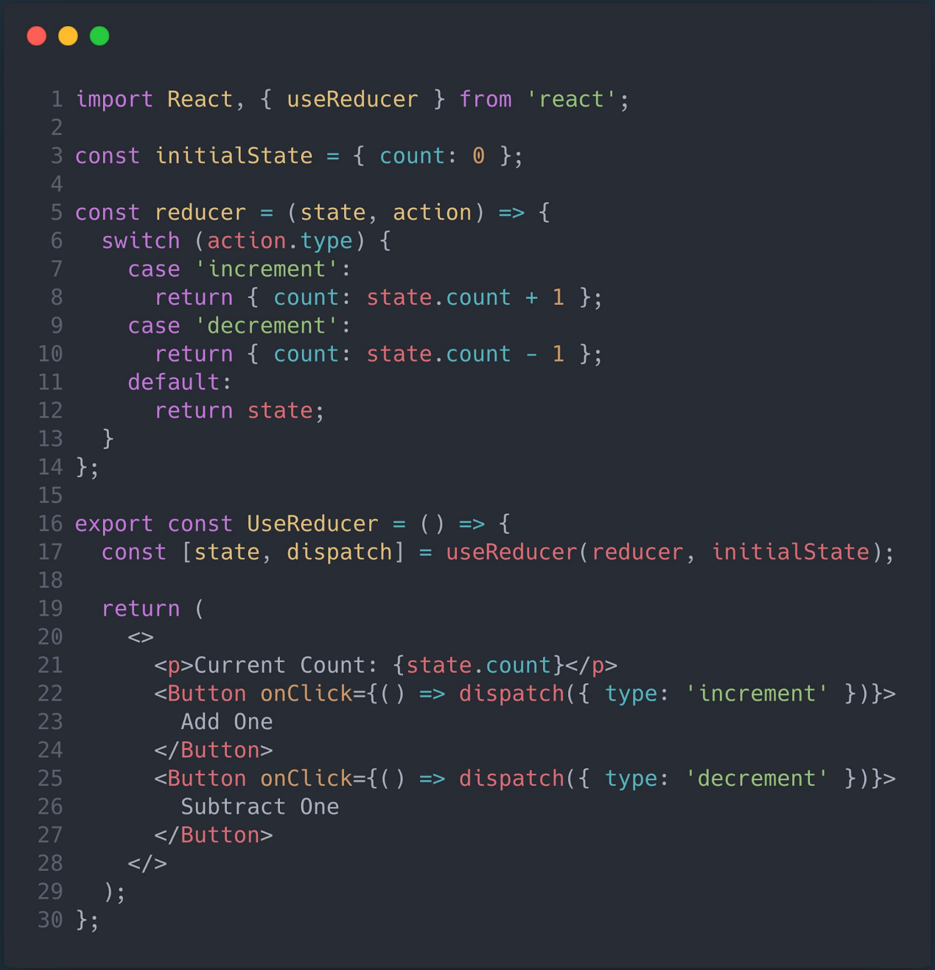


```
let subscription;

useEffect(() => {
  subscription = props.source.subscribe();
}, [props.source])
```



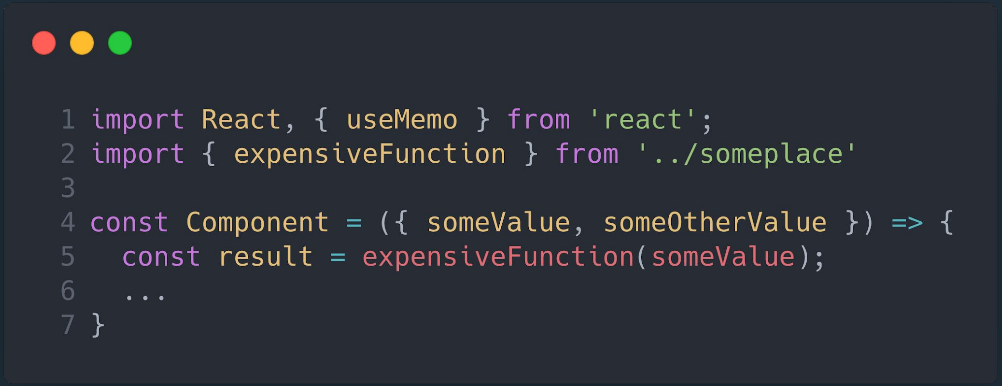
```
let subscription;

useEffect(() => {
  subscription = props.source.subscribe();
  return () => {
    subscription.unsubscribe();
  }
}, [props.source]);
```

useReducer

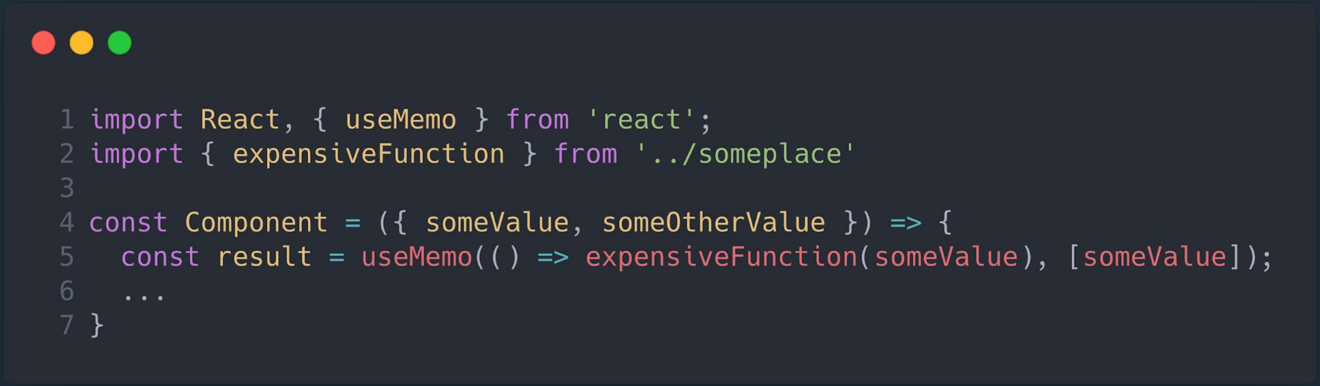


```jsx
import React, { useReducer } from 'react';

const initialState = { count: 0 };

const reducer = (state, action) => {
  switch (action.type) {
    case 'increment':
      return { count: state.count + 1 };
    case 'decrement':
      return { count: state.count - 1 };
    default:
      return state;
  }
};

export const UseReducer = () => {
  const [state, dispatch] = useReducer(reducer, initialState);

  return (
    <>
      <p>Current Count: {state.count}</p>
      <Button onClick={() => dispatch({ type: 'increment' })}>
        Add One
      </Button>
      <Button onClick={() => dispatch({ type: 'decrement' })}>
        Subtract One
      </Button>
    </>
  );
};
```

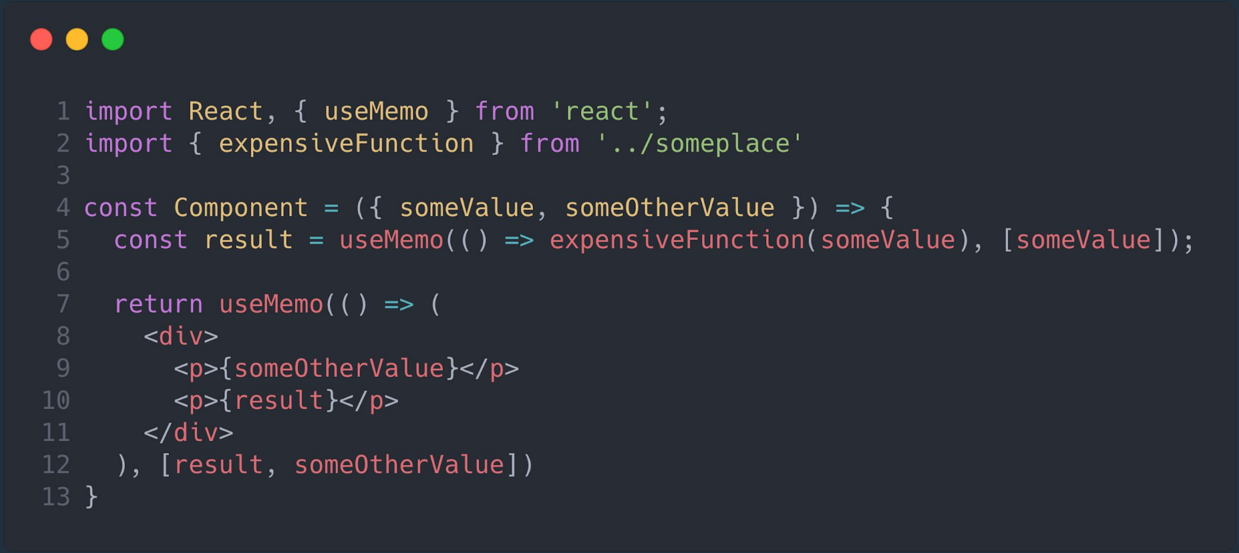
useMemo



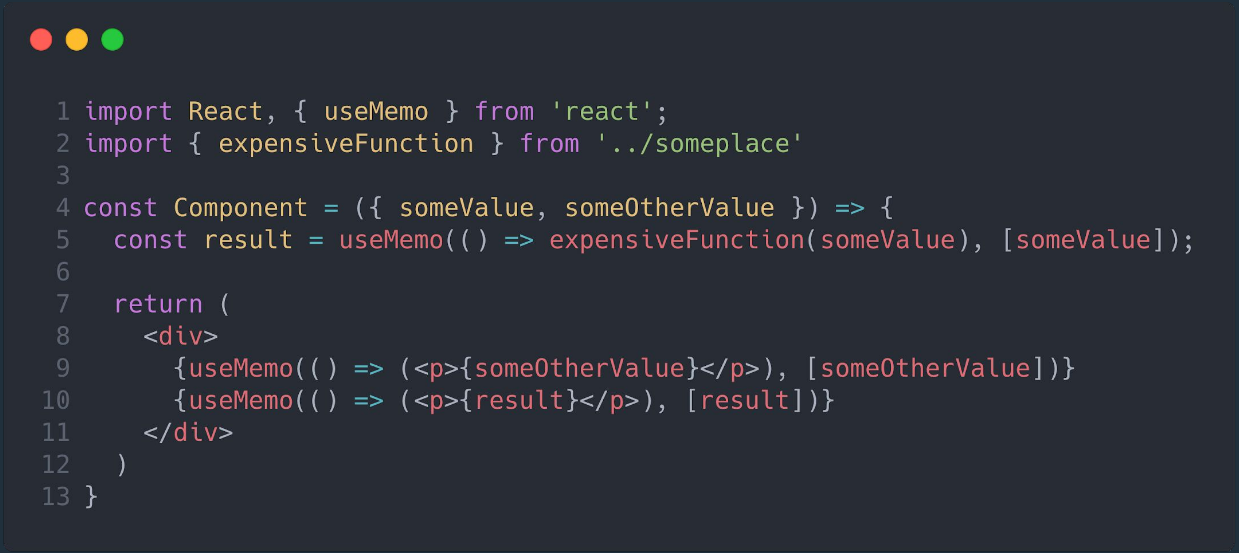
```
import React, { useMemo } from 'react';
import { expensiveFunction } from '../someplace'

const Component = ({ someValue, someOtherValue }) => {
  const result = expensiveFunction(someValue);
  ...
}
```



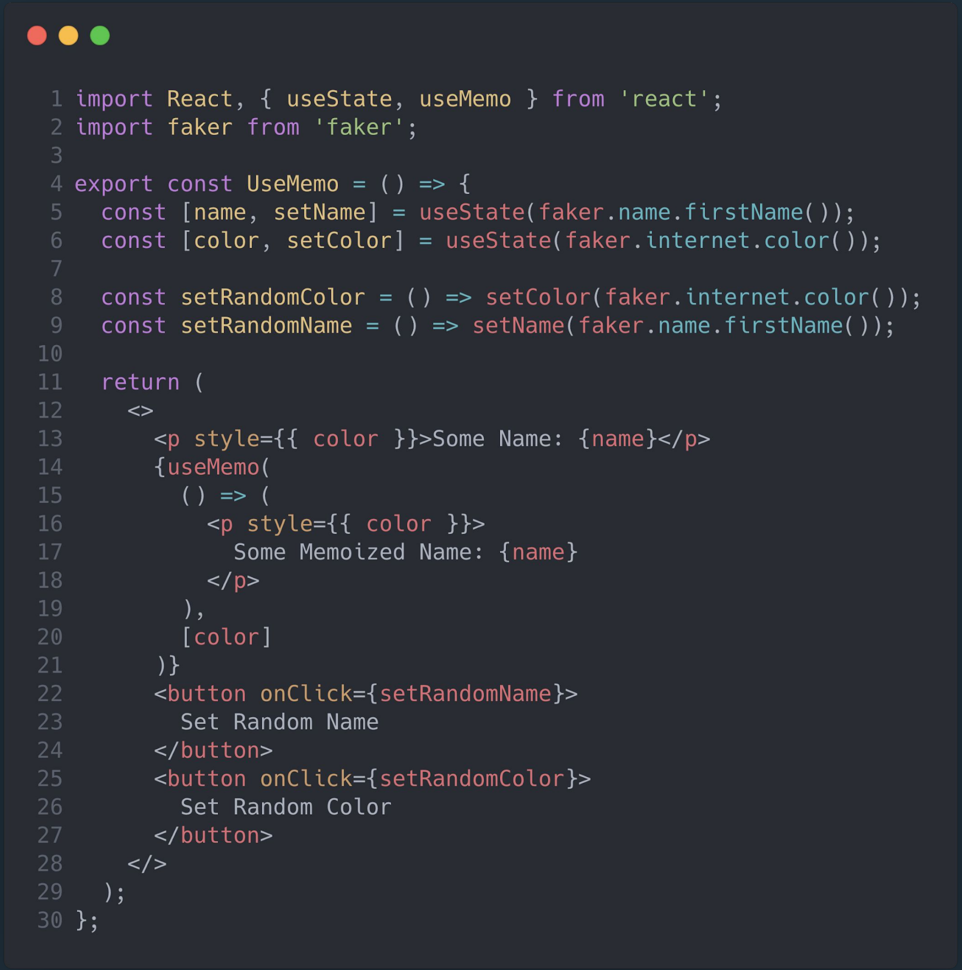
```javascript
import React, { useMemo } from 'react';
import { expensiveFunction } from '../someplace'

const Component = ({ someValue, someOtherValue }) => {
  const result = useMemo(() => expensiveFunction(someValue), [someValue]);
  ...
}
```



```jsx
import React, { useMemo } from 'react';
import { expensiveFunction } from '../someplace'

const Component = ({ someValue, someOtherValue }) => {
  const result = useMemo(() => expensiveFunction(someValue), [someValue]);

  return useMemo(() => (
    <div>
      <p>{someOtherValue}</p>
      <p>{result}</p>
    </div>
  ), [result, someOtherValue])
}
```

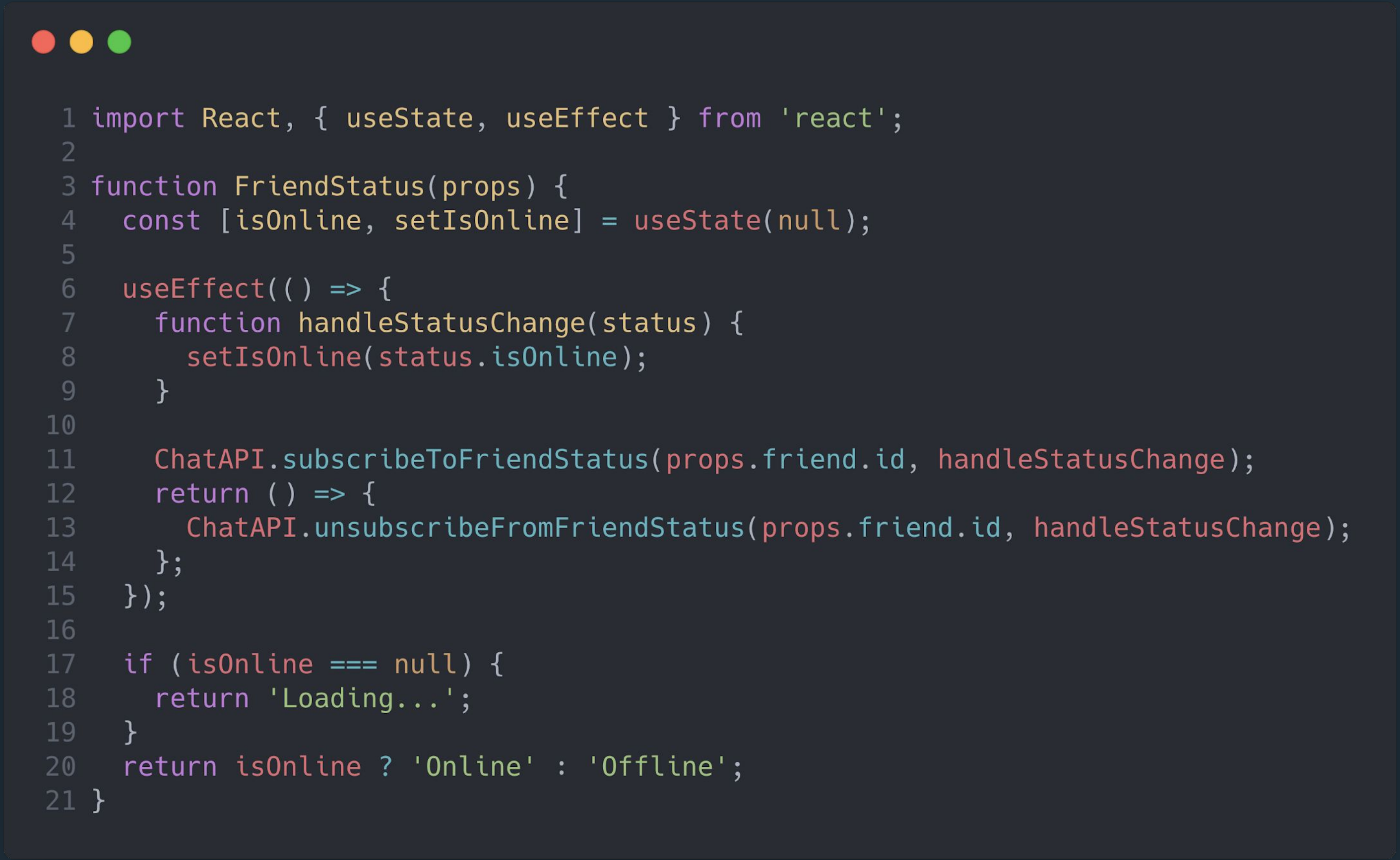


```jsx
import React, { useMemo } from 'react';
import { expensiveFunction } from '../someplace'

const Component = ({ someValue, someOtherValue }) => {
  const result = useMemo(() => expensiveFunction(someValue), [someValue]);

  return (
    <div>
      {useMemo(() => (<p>{someOtherValue}</p>), [someOtherValue])}
      {useMemo(() => (<p>{result}</p>), [result])}
    </div>
  )
}
```

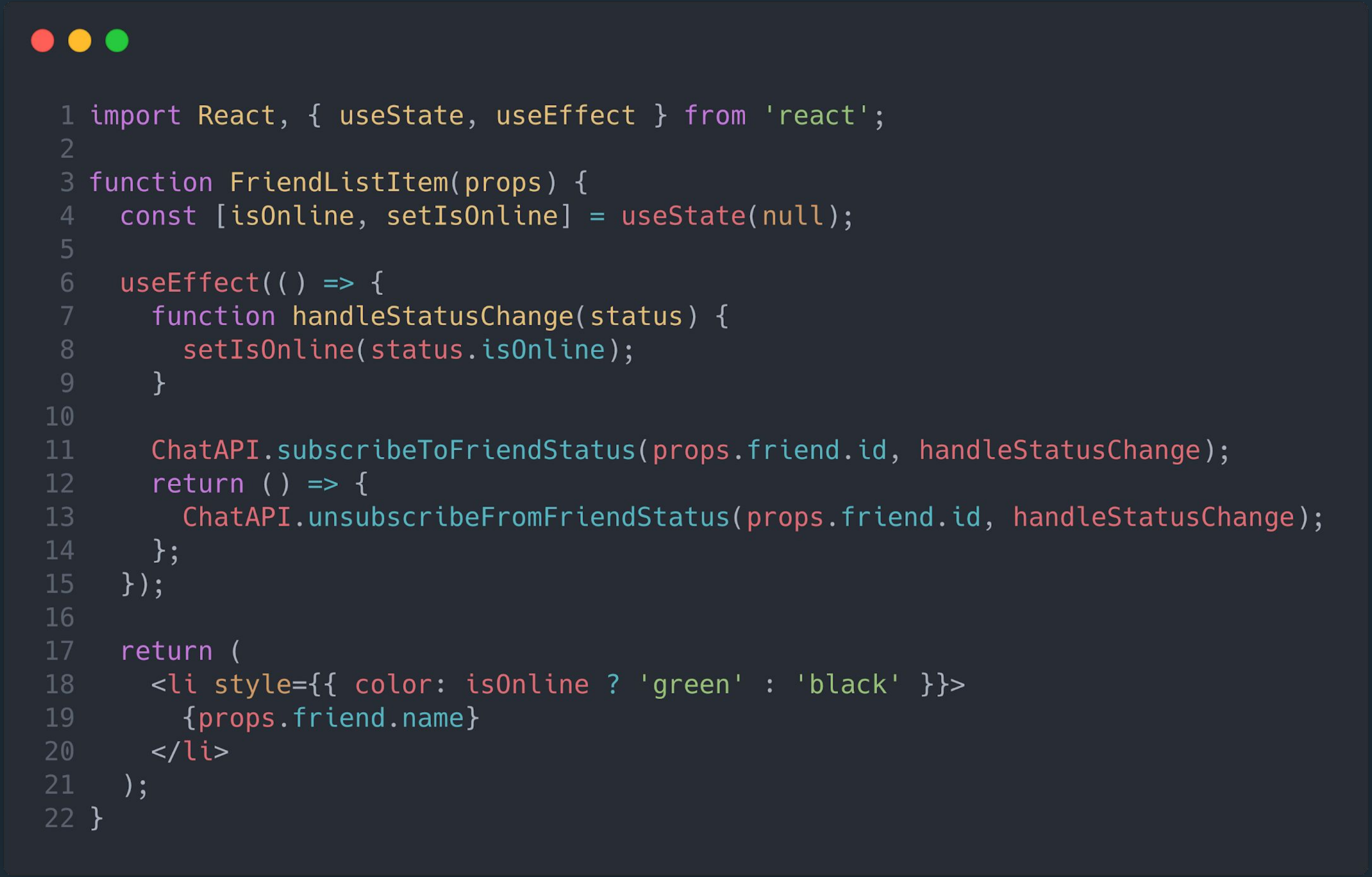


```
import React, { useState, useMemo } from 'react';
import faker from 'faker';

export const UseMemo = () => {
  const [name, setName] = useState(faker.name.firstName());
  const [color, setColor] = useState(faker.internet.color());

  const setRandomColor = () => setColor(faker.internet.color());
  const setRandomName = () => setName(faker.name.firstName());

  return (
    <>
      <p style={{ color }}>Some Name: {name}</p>
      {useMemo(
        () => (
          <p style={{ color }}>
            Some Memoized Name: {name}
          </p>
        ),
        [color]
      )}
      <button onClick={setRandomName}>
        Set Random Name
      </button>
      <button onClick={setRandomColor}>
        Set Random Color
      </button>
    </>
  );
};
```

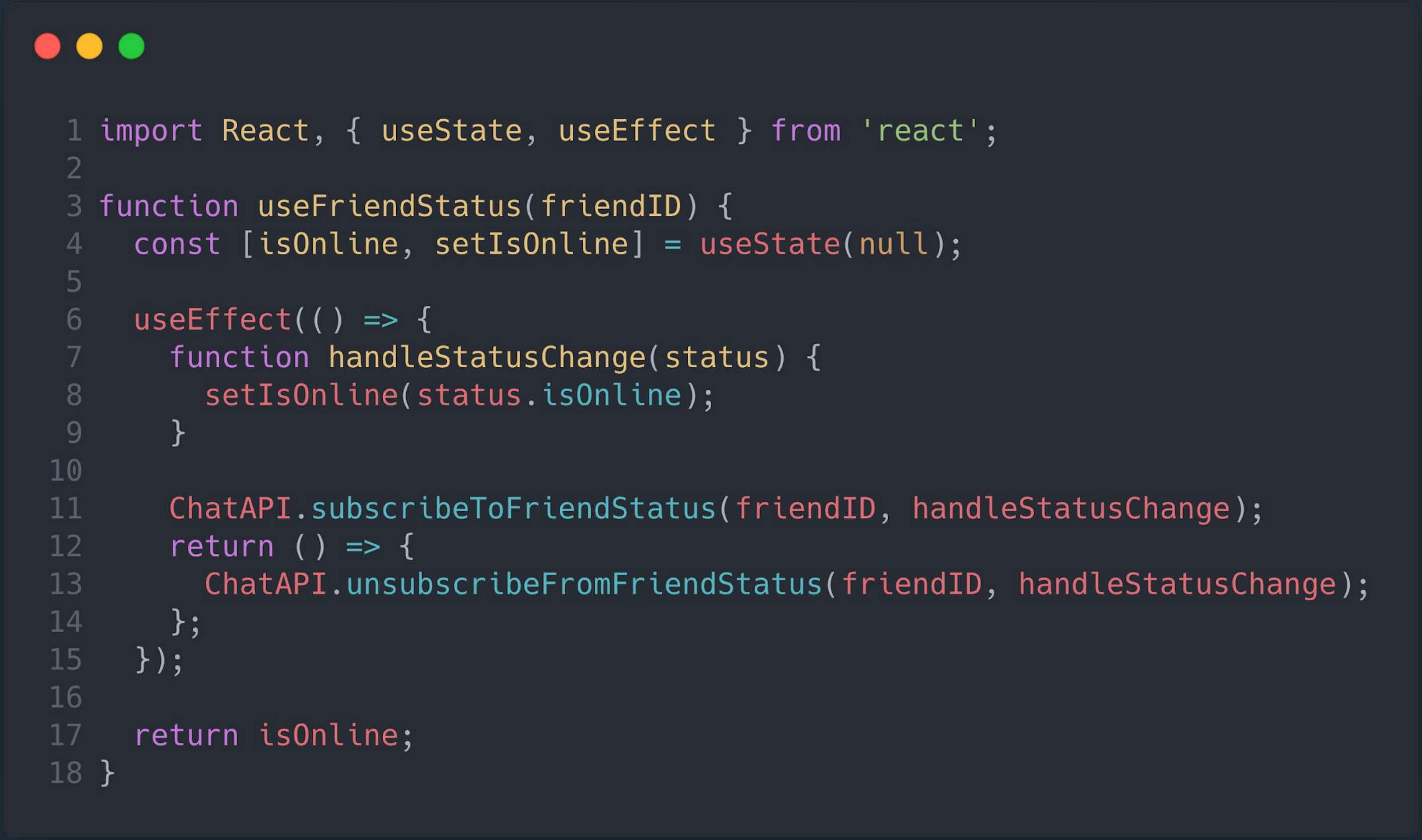
Custom Hooks



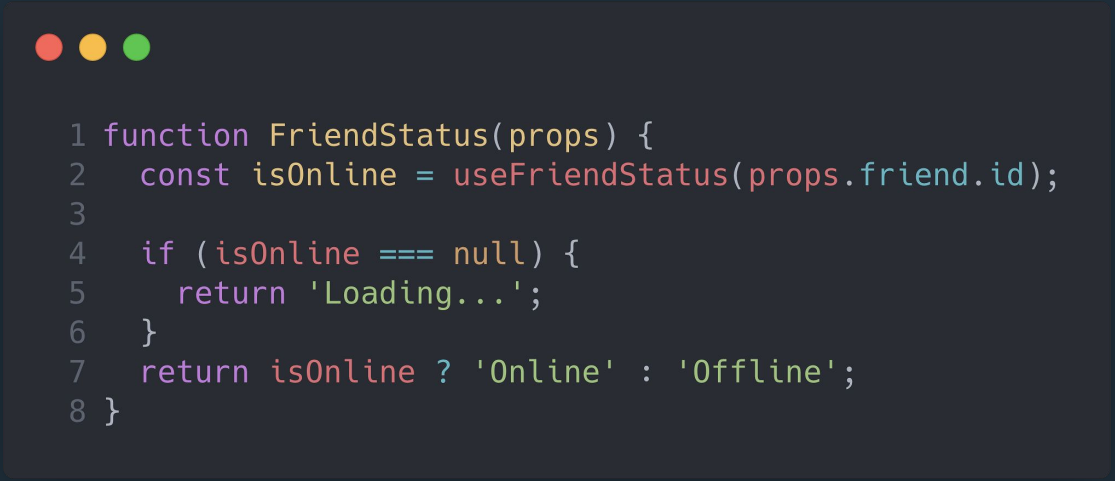
```javascript
import React, { useState, useEffect } from 'react';

function FriendStatus(props) {
  const [isOnline, setIsOnline] = useState(null);

  useEffect(() => {
    function handleStatusChange(status) {
      setIsOnline(status.isOnline);
    }

    ChatAPI.subscribeToFriendStatus(props.friend.id, handleStatusChange);
    return () => {
      ChatAPI.unsubscribeFromFriendStatus(props.friend.id, handleStatusChange);
    };
  });

  if (isOnline === null) {
    return 'Loading...';
  }
  return isOnline ? 'Online' : 'Offline';
}
```



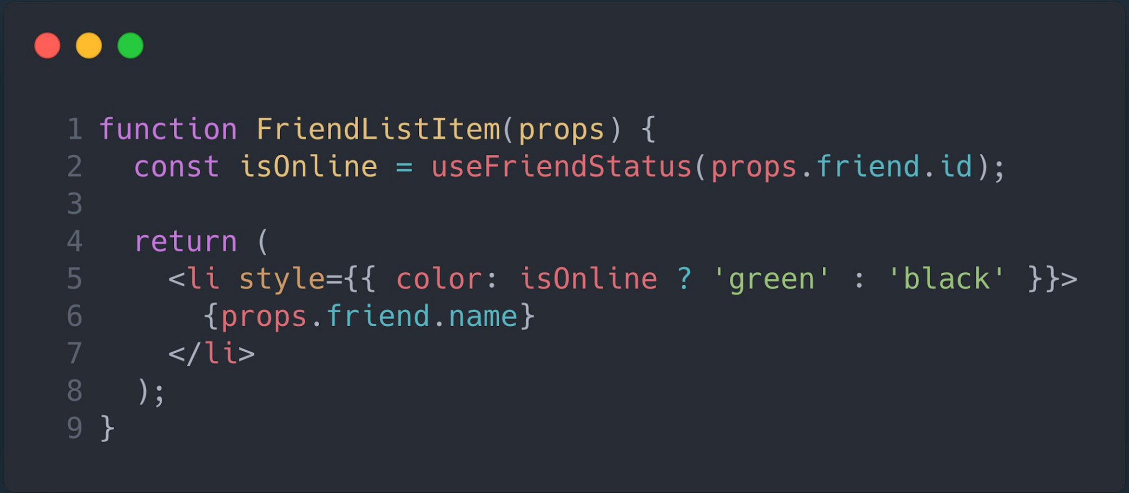
```
import React, { useState, useEffect } from 'react';

function FriendListItem(props) {
  const [isOnline, setIsOnline] = useState(null);

  useEffect(() => {
    function handleStatusChange(status) {
      setIsOnline(status.isOnline);
    }

    ChatAPI.subscribeToFriendStatus(props.friend.id, handleStatusChange);
    return () => {
      ChatAPI.unsubscribeFromFriendStatus(props.friend.id, handleStatusChange);
    };
  });

  return (
    <li style={{ color: isOnline ? 'green' : 'black' }}>
      {props.friend.name}
    </li>
  );
}
```



```javascript
import React, { useState, useEffect } from 'react';

function useFriendStatus(friendID) {
  const [isOnline, setIsOnline] = useState(null);

  useEffect(() => {
    function handleStatusChange(status) {
      setIsOnline(status.isOnline);
    }

    ChatAPI.subscribeToFriendStatus(friendID, handleStatusChange);
    return () => {
      ChatAPI.unsubscribeFromFriendStatus(friendID, handleStatusChange);
    };
  });

  return isOnline;
}
```

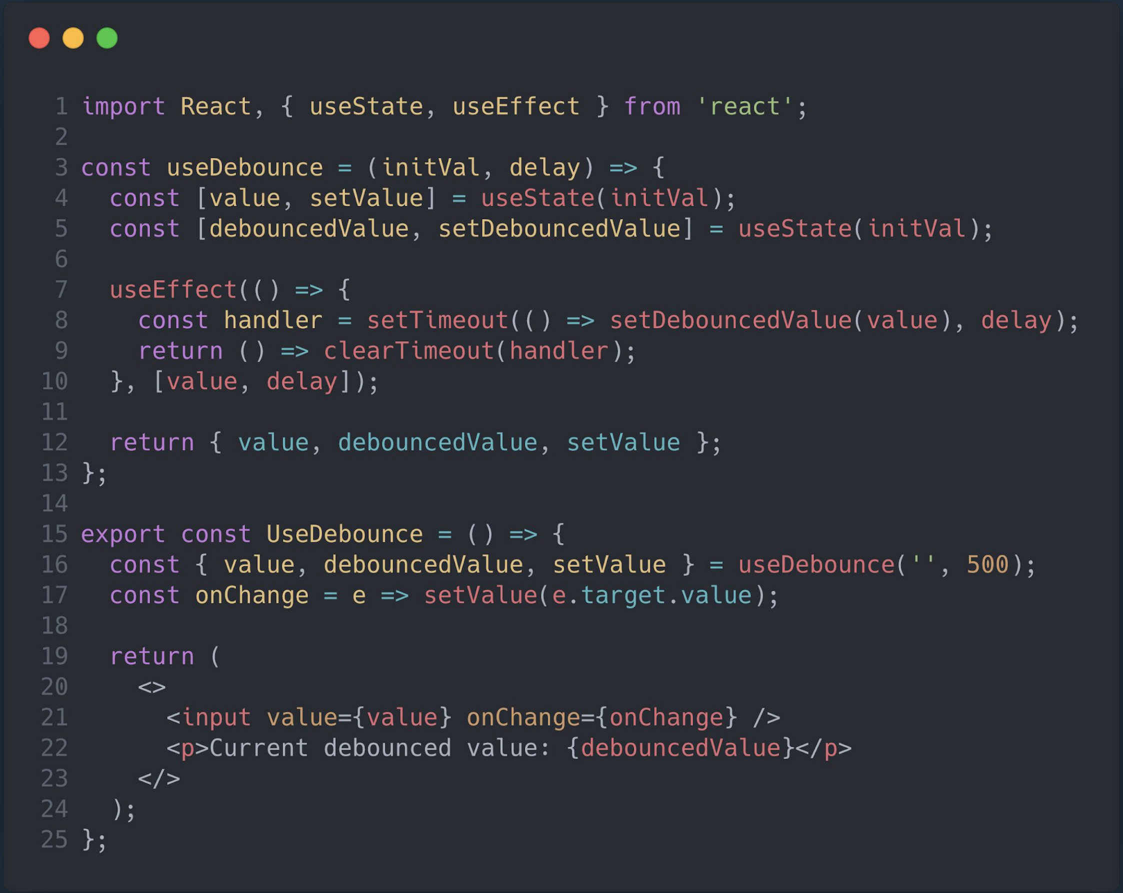


```
function FriendStatus(props) {
  const isOnline = useFriendStatus(props.friend.id);

  if (isOnline === null) {
    return 'Loading...';
  }
  return isOnline ? 'Online' : 'Offline';
}
```



```
function FriendListItem(props) {
  const isOnline = useFriendStatus(props.friend.id);

  return (
    <li style={{ color: isOnline ? 'green' : 'black' }}>
      {props.friend.name}
    </li>
  );
}
```

useDebounce



```
import React, { useState, useEffect } from 'react';

const useDebounce = (initVal, delay) => {
  const [value, setValue] = useState(initVal);
  const [debouncedValue, setDebouncedValue] = useState(initVal);

  useEffect(() => {
    const handler = setTimeout(() => setDebouncedValue(value), delay);
    return () => clearTimeout(handler);
  }, [value, delay]);

  return { value, debouncedValue, setValue };
};

export const UseDebounce = () => {
  const { value, debouncedValue, setValue } = useDebounce('', 500);
  const onChange = e => setValue(e.target.value);

  return (
    <>
      <input value={value} onChange={onChange} />
      <p>Current debounced value: {debouncedValue}</p>
    </>
  );
};
```

- 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

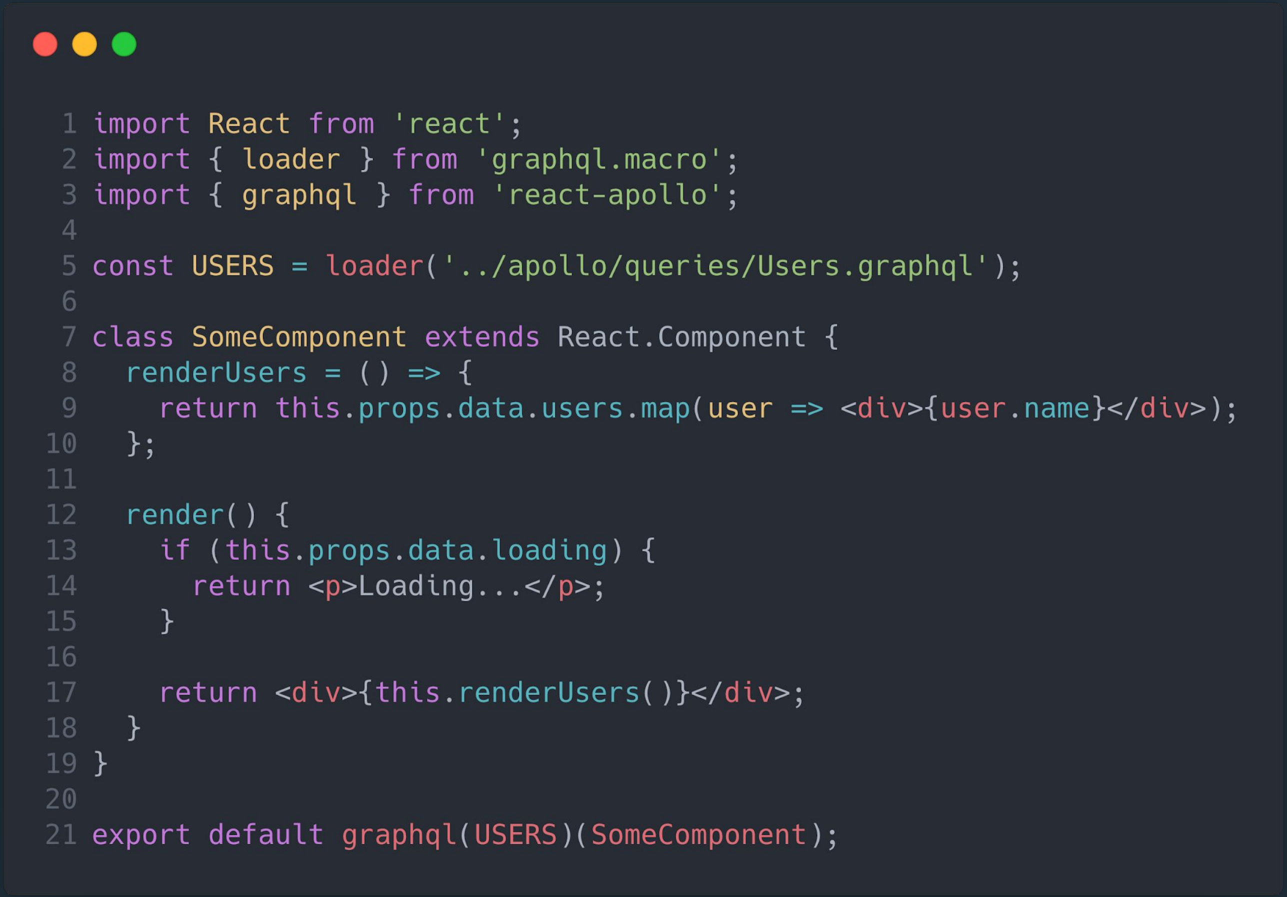
Our own Redux

- 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

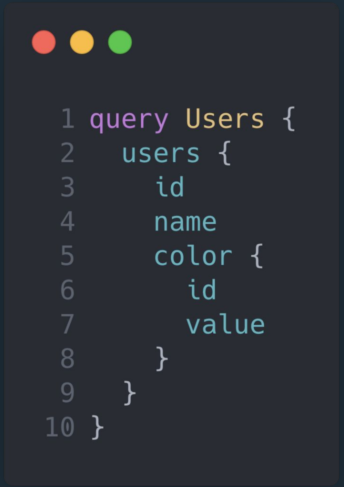
03

Hooks and GraphQL.

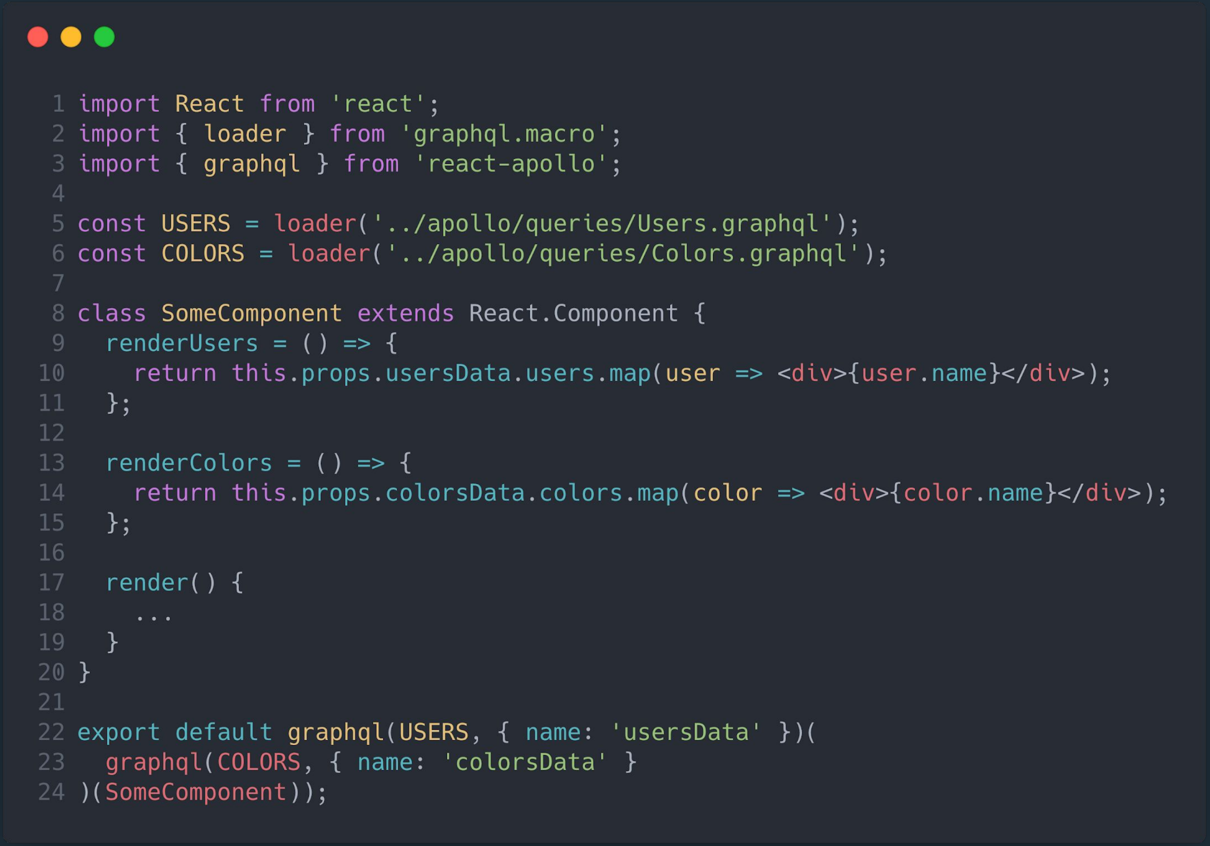
Apollo without hooks



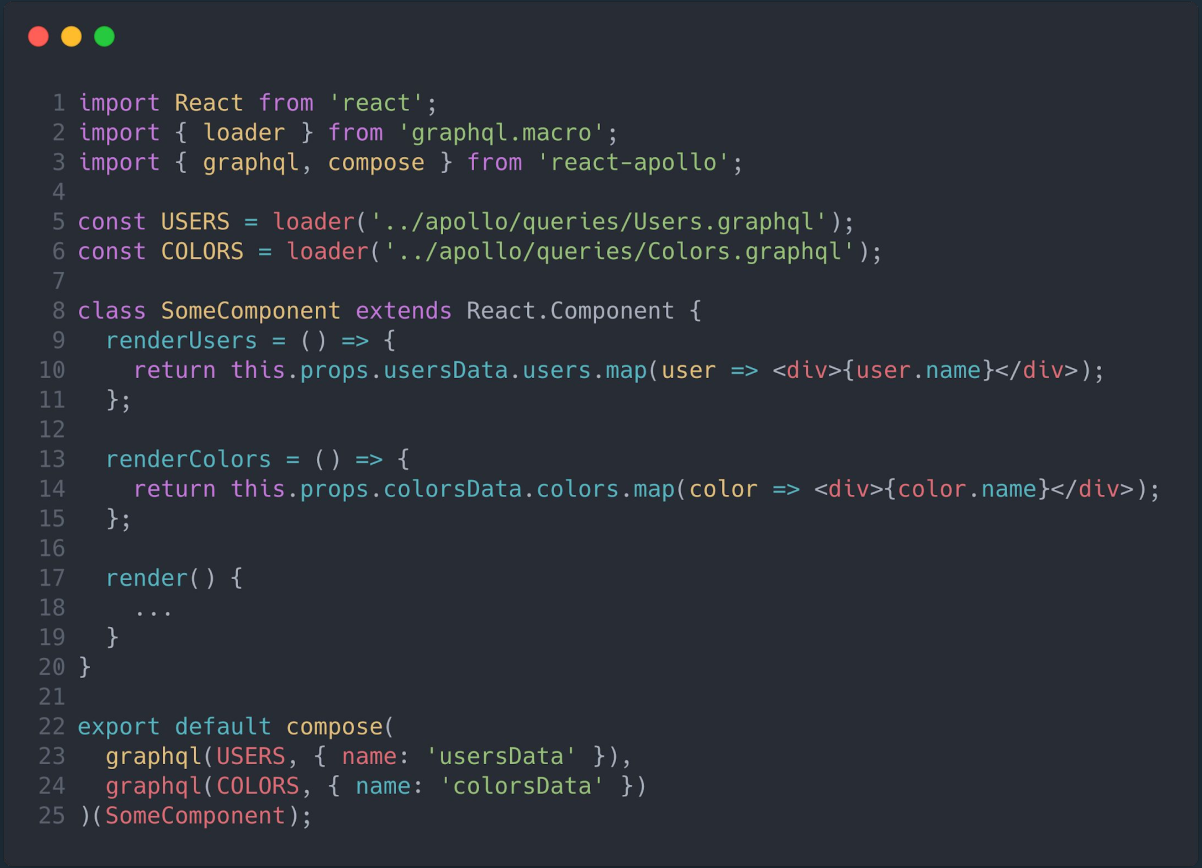
```
import React from 'react';
import { loader } from 'graphql.macro';
import { graphql } from 'react-apollo';

const USERS = loader('../apollo/queries/Users.graphql');

class SomeComponent extends React.Component {
  renderUsers = () => {
    return this.props.data.users.map(user => <div>{user.name}</div>);
  };

  render() {
    if (this.props.data.loading) {
      return <p>Loading...</p>;
    }

    return <div>{this.renderUsers()}</div>;
  }
}

export default graphql(USERS)(SomeComponent);
```



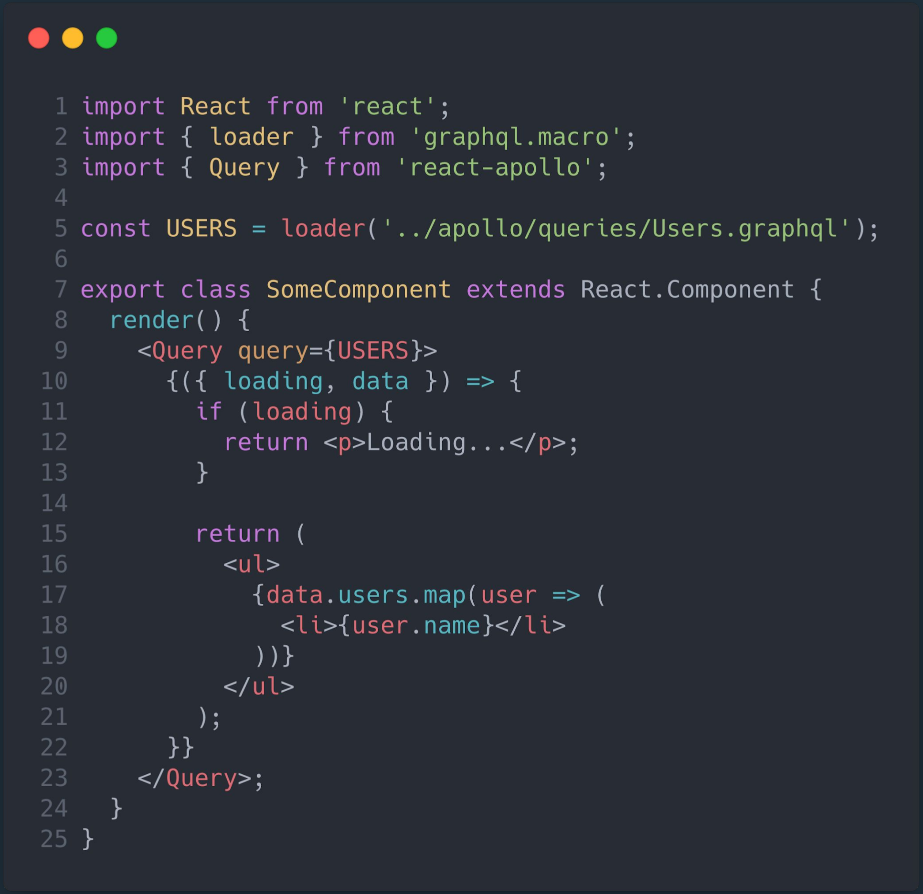
```graphql
query Users {
  users {
    id
    name
    color {
      id
      value
    }
  }
}
```



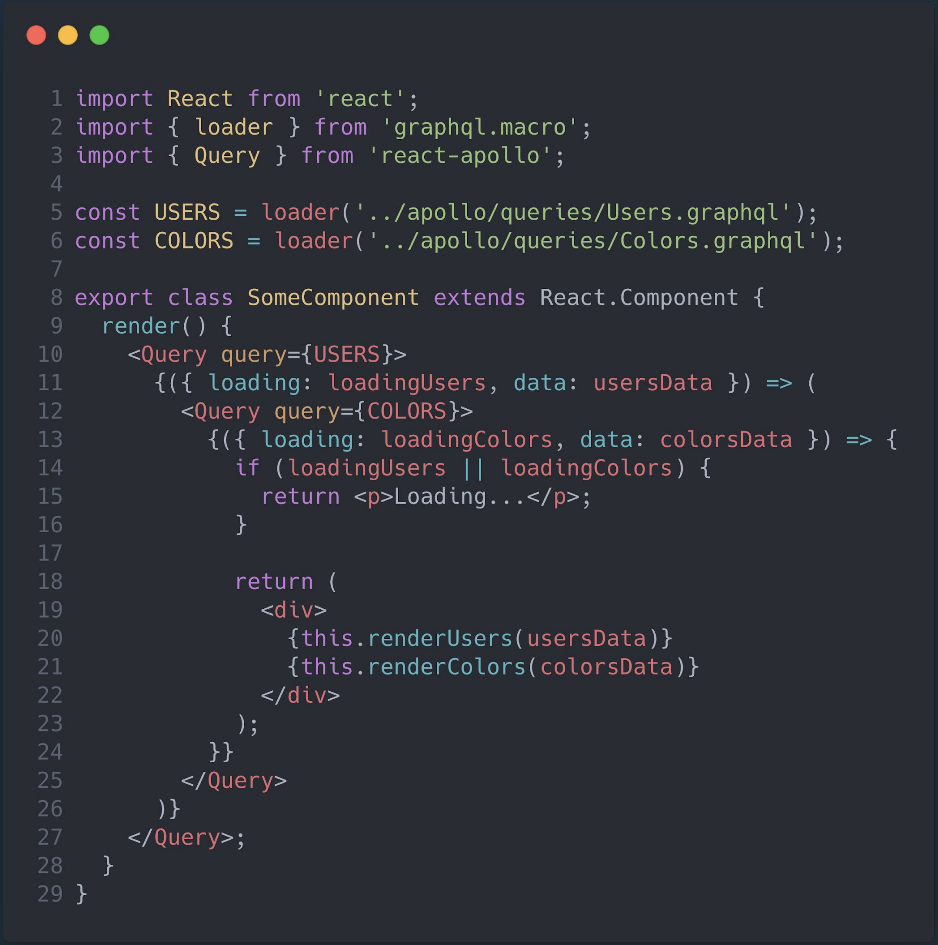
```
import React from 'react';
import { loader } from 'graphql.macro';
import { graphql } from 'react-apollo';

const USERS = loader('../apollo/queries/Users.graphql');
const COLORS = loader('../apollo/queries/Colors.graphql');

class SomeComponent extends React.Component {
  renderUsers = () => {
    return this.props.usersData.users.map(user => <div>{user.name}</div>);
  };

  renderColors = () => {
    return this.props.colorsData.colors.map(color => <div>{color.name}</div>);
  };

  render() {
    ...
  }
}

export default graphql(USERS, { name: 'usersData' })(
  graphql(COLORS, { name: 'colorsData' }
)(SomeComponent));
```



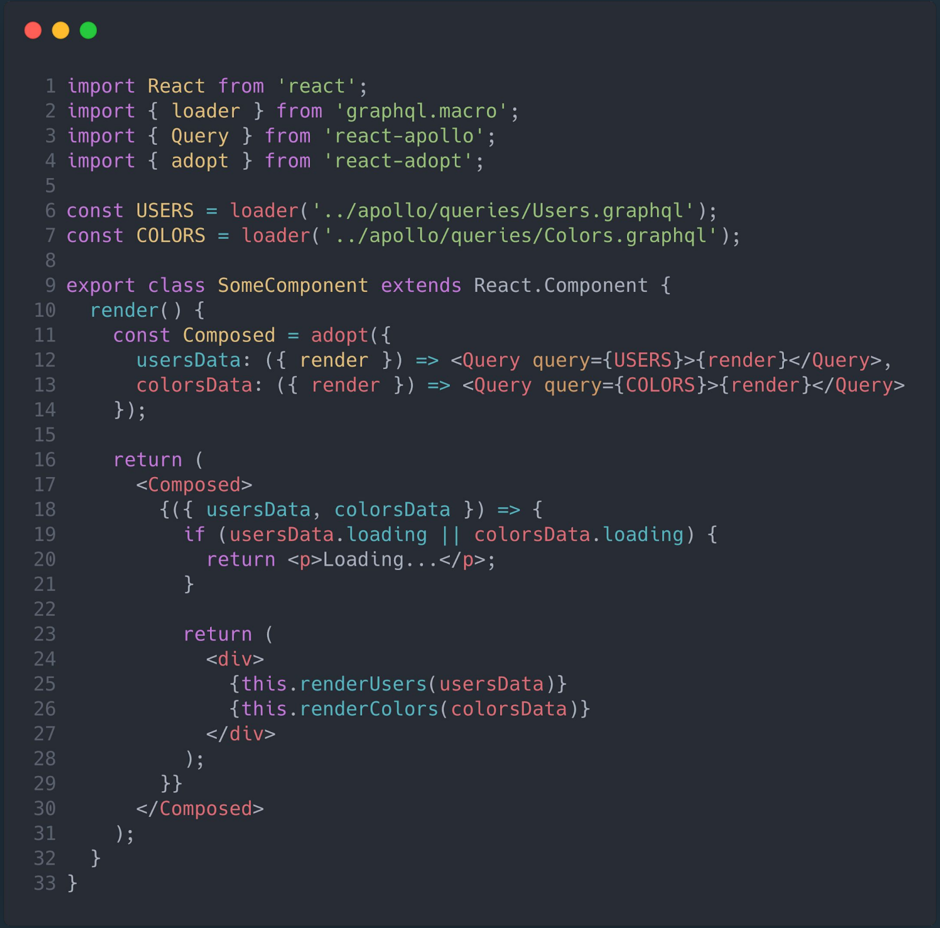
```
import React from 'react';
import { loader } from 'graphql.macro';
import { graphql, compose } from 'react-apollo';

const USERS = loader('../apollo/queries/Users.graphql');
const COLORS = loader('../apollo/queries/Colors.graphql');

class SomeComponent extends React.Component {
  renderUsers = () => {
    return this.props.usersData.users.map(user => <div>{user.name}</div>);
  };

  renderColors = () => {
    return this.props.colorsData.colors.map(color => <div>{color.name}</div>);
  };

  render() {
    ...
  }
}

export default compose(
  graphql(USERS, { name: 'usersData' }),
  graphql(COLORS, { name: 'colorsData' })
)(SomeComponent);
```



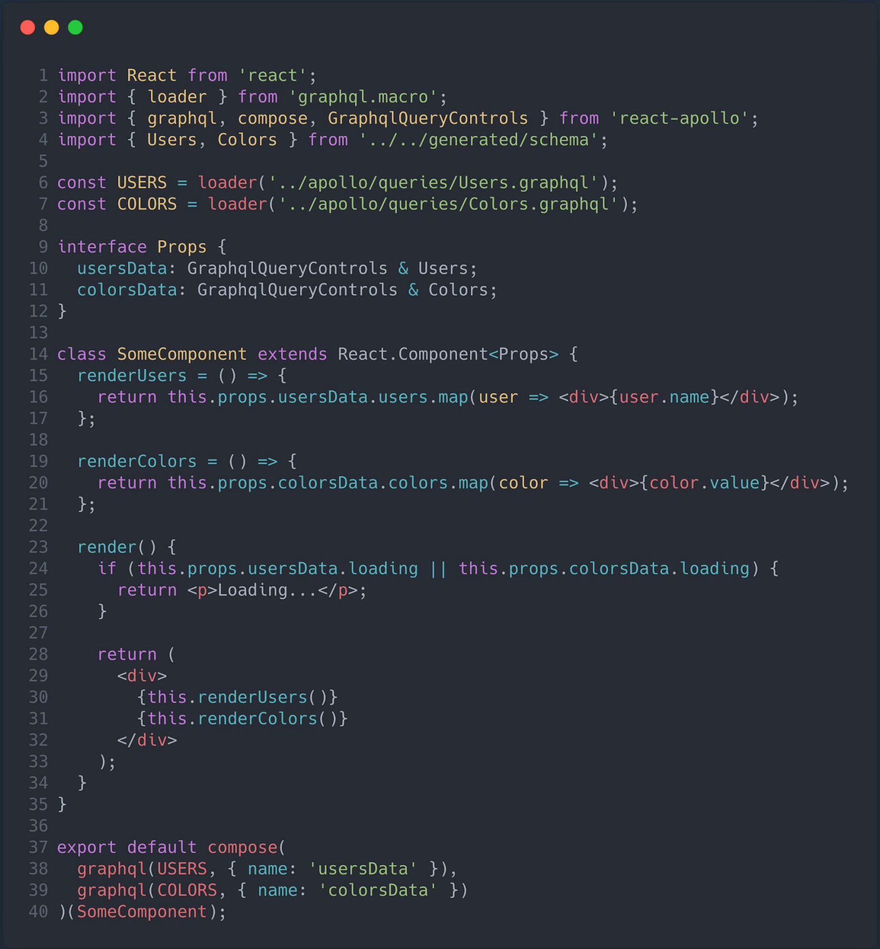
```
import React from 'react';
import { loader } from 'graphql.macro';
import { Query } from 'react-apollo';

const USERS = loader('../apollo/queries/Users.graphql');

export class SomeComponent extends React.Component {
  render() {
    <Query query={USERS}>
      {({ loading, data }) => {
        if (loading) {
          return <p>Loading...</p>;
        }

        return (
          <ul>
            {data.users.map(user => (
              <li>{user.name}</li>
            ))}
          </ul>
        );
      }}
    </Query>;
  }
}
```



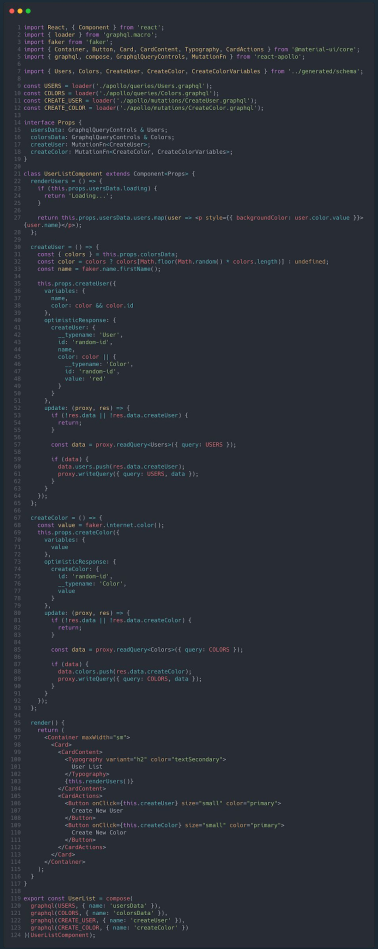
```
import React from 'react';
import { loader } from 'graphql.macro';
import { Query } from 'react-apollo';

const USERS = loader('../apollo/queries/Users.graphql');
const COLORS = loader('../apollo/queries/Colors.graphql');

export class SomeComponent extends React.Component {
  render() {
    <Query query={USERS}>
      {({ loading: loadingUsers, data: usersData }) => (
        <Query query={COLORS}>
          {({ loading: loadingColors, data: colorsData }) => {
            if (loadingUsers || loadingColors) {
              return <p>Loading...</p>;
            }

            return (
              <div>
                {this.renderUsers(usersData)}
                {this.renderColors(colorsData)}
              </div>
            );
          }}
        </Query>
      )}
    </Query>;
  }
}
```



```
import React from 'react';
import { loader } from 'graphql.macro';
import { Query } from 'react-apollo';
import { adopt } from 'react-adopt';

const USERS = loader('../apollo/queries/Users.graphql');
const COLORS = loader('../apollo/queries/Colors.graphql');

export class SomeComponent extends React.Component {
  render() {
    const Composed = adopt({
      usersData: ({ render }) => <Query query={USERS}>{render}</Query>,
      colorsData: ({ render }) => <Query query={COLORS}>{render}</Query>
    });

    return (
      <Composed>
        {({ usersData, colorsData }) => {
          if (usersData.loading || colorsData.loading) {
            return <p>Loading...</p>;
          }

          return (
            <div>
              {this.renderUsers(usersData)}
              {this.renderColors(colorsData)}
            </div>
          );
        }}
      </Composed>
    );
  }
}
```

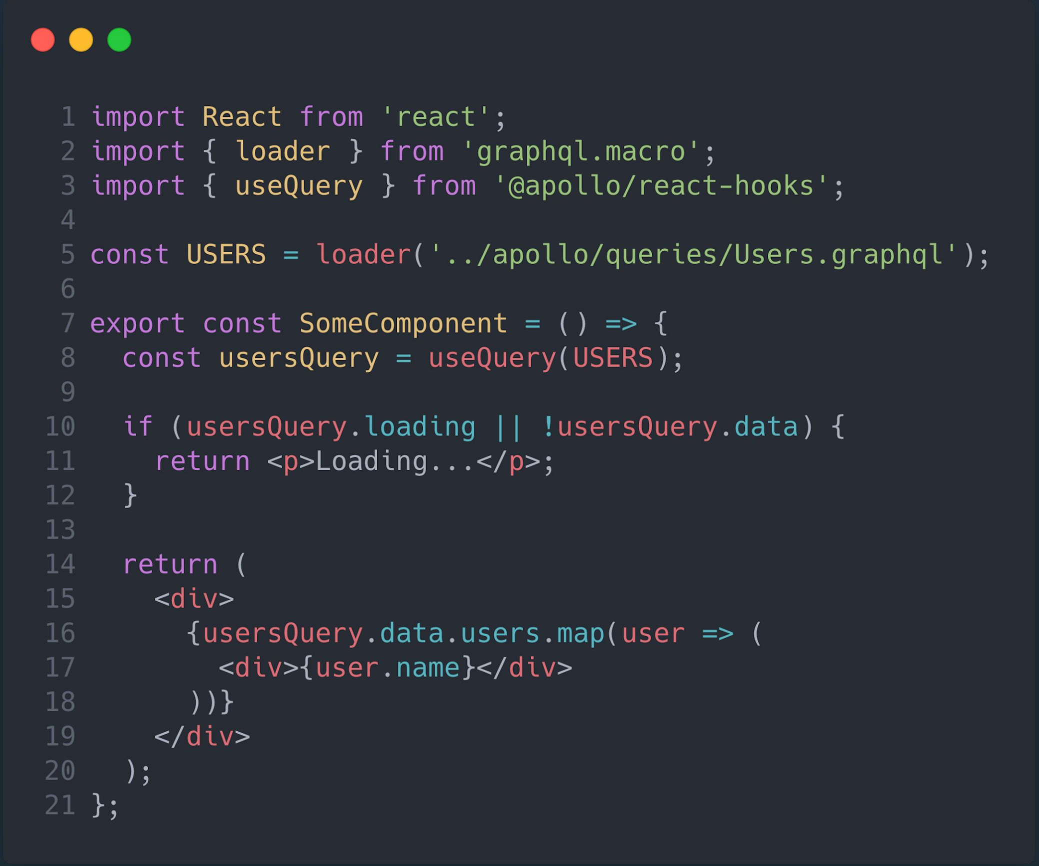


```
import React from 'react';
import { loader } from 'graphql.macro';
import { graphql, compose, GraphqlQueryControls } from 'react-apollo';
import { Users, Colors } from '../../generated/schema';

const USERS = loader('../apollo/queries/Users.graphql');
const COLORS = loader('../apollo/queries/Colors.graphql');

interface Props {
  usersData: GraphqlQueryControls & Users;
  colorsData: GraphqlQueryControls & Colors;
}

class SomeComponent extends React.Component<Props> {
  renderUsers = () => {
    return this.props.usersData.users.map(user => <div>{user.name}</div>);
  };

  renderColors = () => {
    return this.props.colorsData.colors.map(color => <div>{color.value}</div>);
  };

  render() {
    if (this.props.usersData.loading || this.props.colorsData.loading) {
      return <p>Loading...</p>;
    }

    return (
      <div>
        {this.renderUsers()}
        {this.renderColors()}
      </div>
    );
  }
}

export default compose(
  graphql(USERS, { name: 'usersData' }),
  graphql(COLORS, { name: 'colorsData' })
)(SomeComponent);
```

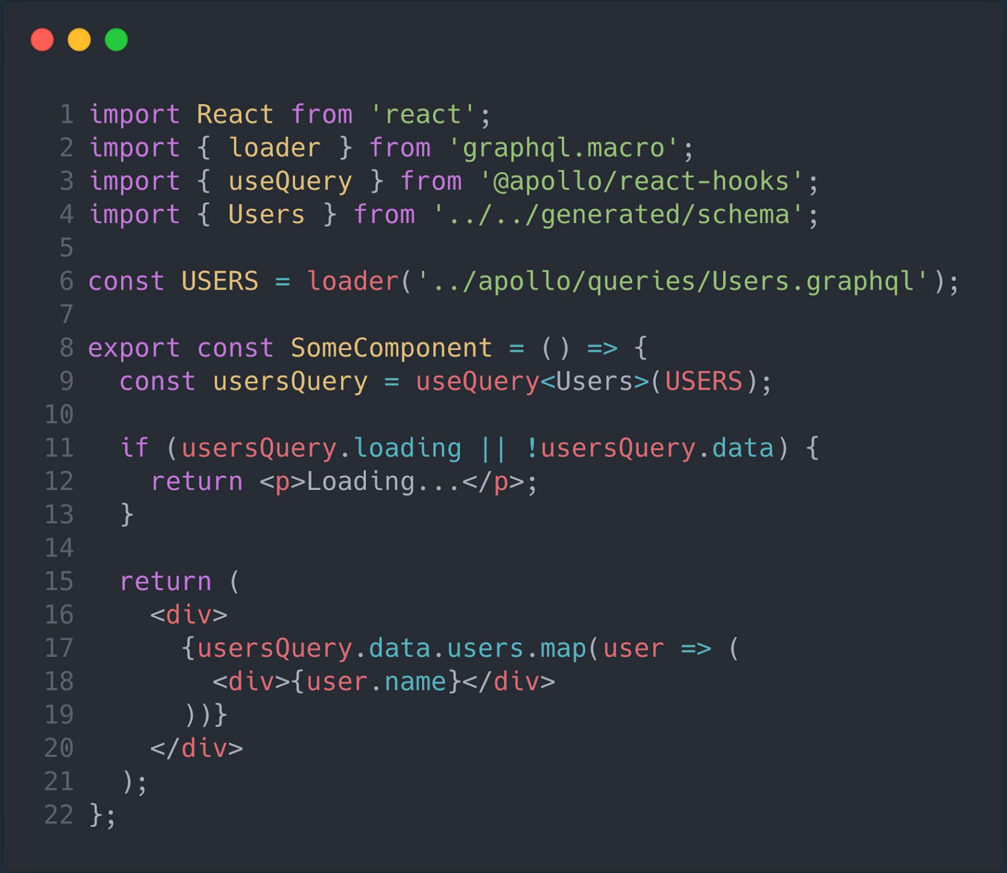


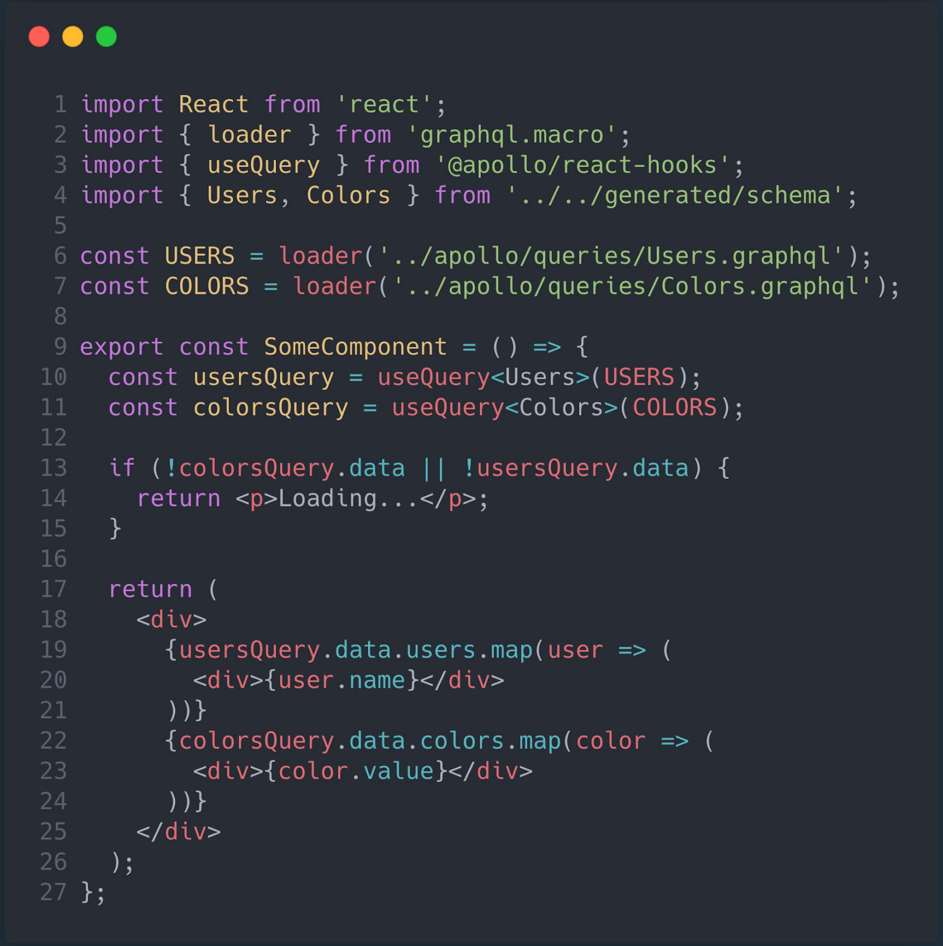
```tsx
import React, { Component } from 'react';
import { loader } from 'graphql.macro';
import faker from 'faker';
import { Container, Button, Card, CardContent, Typography, CardActions } from '@material-ui/core';
import { graphql, compose, GraphqlQueryControls, MutationFn } from 'react-apollo';

import { Users, Colors, CreateUser, CreateColor, CreateColorVariables } from '../generated/schema';

const USERS = loader('./apollo/queries/Users.graphql');
const COLORS = loader('./apollo/queries/Colors.graphql');
const CREATE_USER = loader('./apollo/mutations/CreateUser.graphql');
const CREATE_COLOR = loader('./apollo/mutations/CreateColor.graphql');

interface Props {
  usersData: GraphqlQueryControls & Users;
  colorsData: GraphqlQueryControls & Colors;
  createUser: MutationFn<CreateUser>;
  createColor: MutationFn<CreateColor, CreateColorVariables>;
}

class UserListComponent extends Component<Props> {
  renderUsers = () => {
    if (this.props.usersData.loading) {
      return 'Loading...';
    }

    return this.props.usersData.users.map(user => <p style={{ backgroundColor: user.color.value }}>{user.name}</p>);
  };

  createUser = () => {
    const { colors } = this.props.colorsData;
    const color = colors ? colors[Math.floor(Math.random() * colors.length)] : undefined;
    const name = faker.name.firstName();

    this.props.createUser({
      variables: {
        name,
        color: color && color.id
      },
      optimisticResponse: {
        createUser: {
          __typename: 'User',
          id: 'random-id',
          name,
          color: color || {
            __typename: 'Color',
            id: 'random-id',
            value: 'red'
          }
        }
      },
      update: (proxy, res) => {
        if (!res.data || !res.data.createUser) {
          return;
        }

        const data = proxy.readQuery<Users>({ query: USERS });

        if (data) {
          data.users.push(res.data.createUser);
          proxy.writeQuery({ query: USERS, data });
        }
      }
    });
  };

  createColor = () => {
    const value = faker.internet.color();
    this.props.createColor({
      variables: {
        value
      },
      optimisticResponse: {
        createColor: {
          id: 'random-id',
          __typename: 'Color',
          value
        }
      },
      update: (proxy, res) => {
        if (!res.data || !res.data.createColor) {
          return;
        }

        const data = proxy.readQuery<Colors>({ query: COLORS });

        if (data) {
          data.colors.push(res.data.createColor);
          proxy.writeQuery({ query: COLORS, data });
        }
      }
    });
  };

  render() {
    return (
      <Container maxWidth="sm">
        <Card>
          <CardContent>
            <Typography variant="h2" color="textSecondary">
              User List
            </Typography>
            {this.renderUsers()}
          </CardContent>
          <CardActions>
            <Button onClick={this.createUser} size="small" color="primary">
              Create New User
            </Button>
            <Button onClick={this.createColor} size="small" color="primary">
              Create New Color
            </Button>
          </CardActions>
        </Card>
      </Container>
    );
  }
}

export const UserList = compose(
  graphql(USERS, { name: 'usersData' }),
  graphql(COLORS, { name: 'colorsData' }),
  graphql(CREATE_USER, { name: 'createUser' }),
  graphql(CREATE_COLOR, { name: 'createColor' })
)(UserListComponent);
```

With Hooks

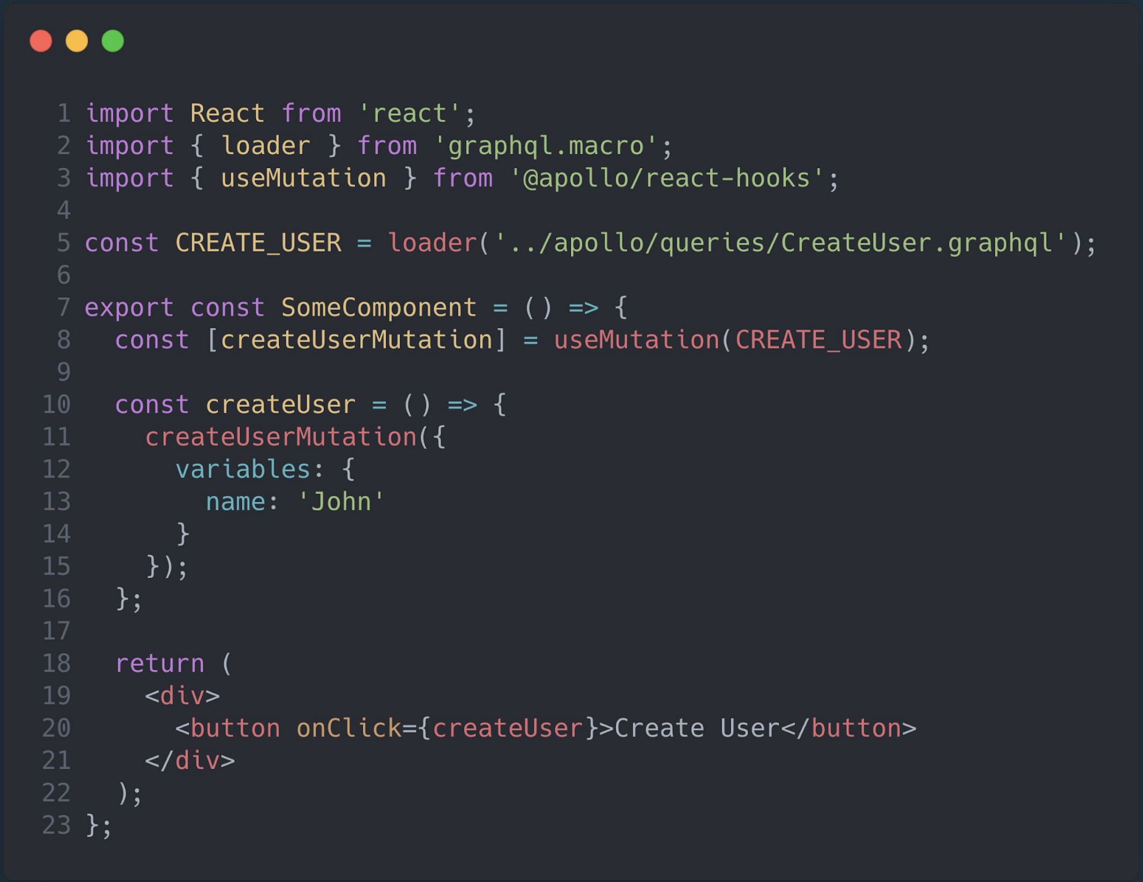


```
import React from 'react';
import { loader } from 'graphql.macro';
import { useQuery } from '@apollo/react-hooks';

const USERS = loader('../apollo/queries/Users.graphql');

export const SomeComponent = () => {
  const usersQuery = useQuery(USERS);

  if (usersQuery.loading || !usersQuery.data) {
    return <p>Loading...</p>;
  }

  return (
    <div>
      {usersQuery.data.users.map(user => (
        <div>{user.name}</div>
      ))}
    </div>
  );
};
```

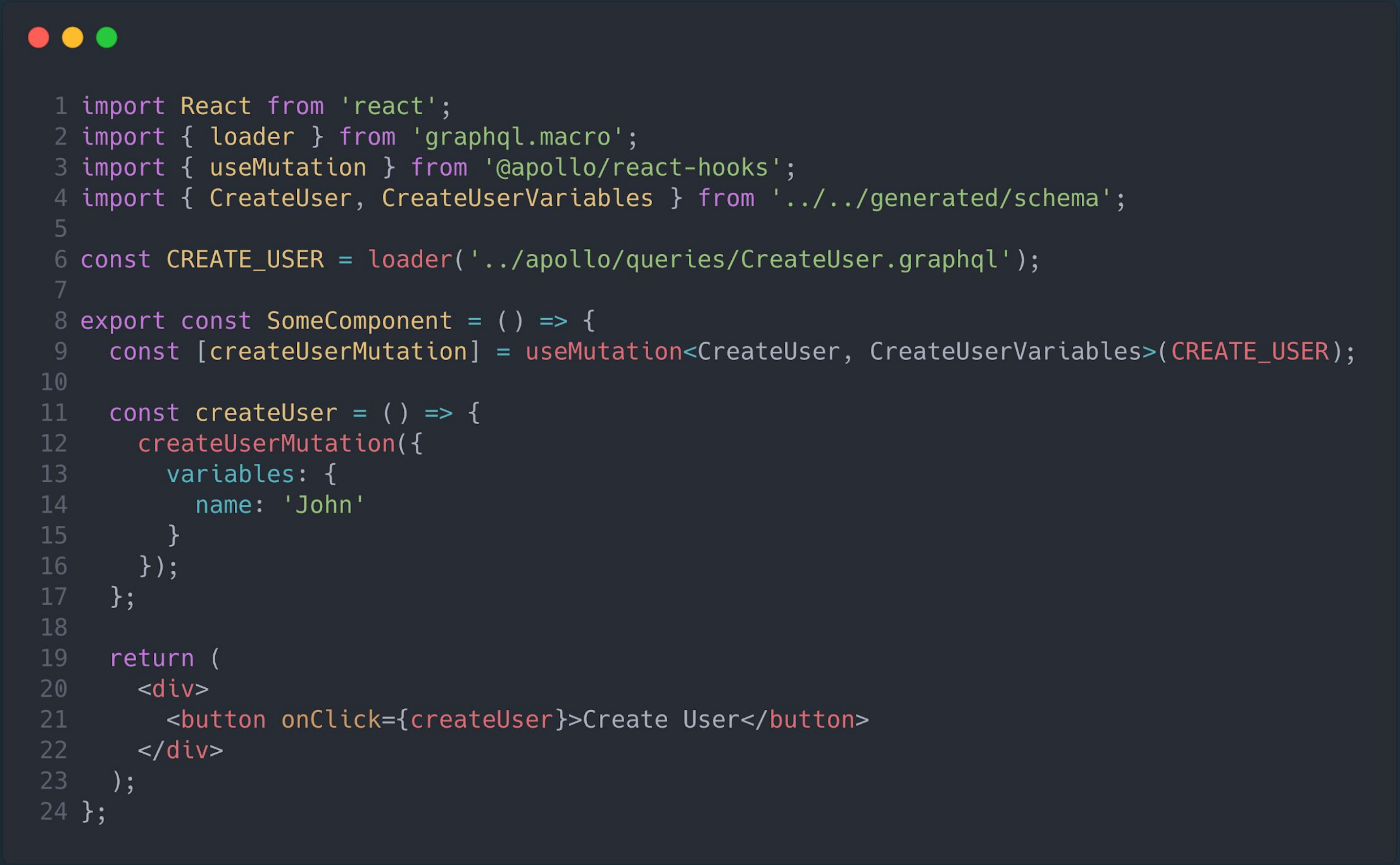

```
import React from 'react';
import { loader } from 'graphql.macro';
import { useQuery } from '@apollo/react-hooks';
import { Users } from '../../generated/schema';

const USERS = loader('../apollo/queries/Users.graphql');

export const SomeComponent = () => {
  const usersQuery = useQuery<Users>(USERS);

  if (usersQuery.loading || !usersQuery.data) {
    return <p>Loading...</p>;
  }

  return (
    <div>
      {usersQuery.data.users.map(user => (
        <div>{user.name}</div>
      ))}
    </div>
  );
};
```



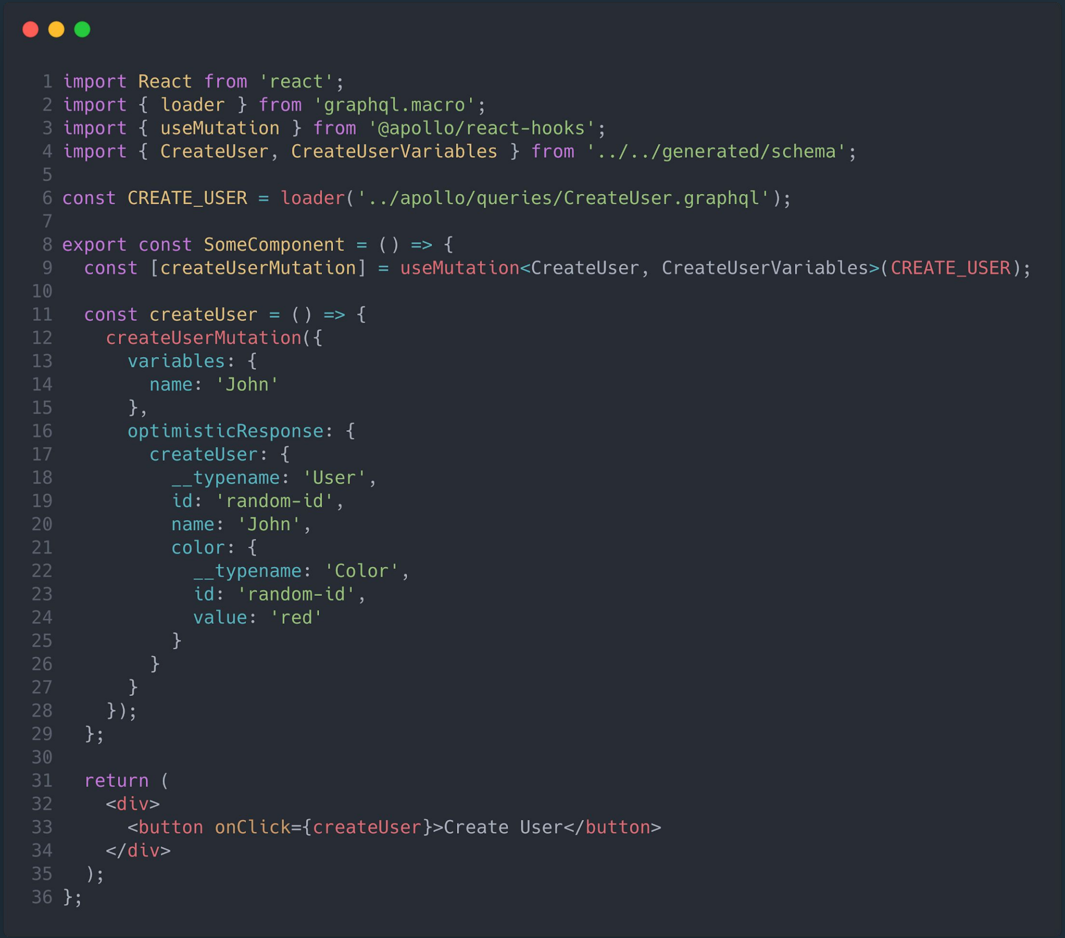
```
import React from 'react';
import { loader } from 'graphql.macro';
import { useQuery } from '@apollo/react-hooks';
import { Users, Colors } from '../../generated/schema';

const USERS = loader('../apollo/queries/Users.graphql');
const COLORS = loader('../apollo/queries/Colors.graphql');

export const SomeComponent = () => {
  const usersQuery = useQuery<Users>(USERS);
  const colorsQuery = useQuery<Colors>(COLORS);

  if (!colorsQuery.data || !usersQuery.data) {
    return <p>Loading...</p>;
  }

  return (
    <div>
      {usersQuery.data.users.map(user => (
        <div>{user.name}</div>
      ))}
      {colorsQuery.data.colors.map(color => (
        <div>{color.value}</div>
      ))}
    </div>
  );
};
```



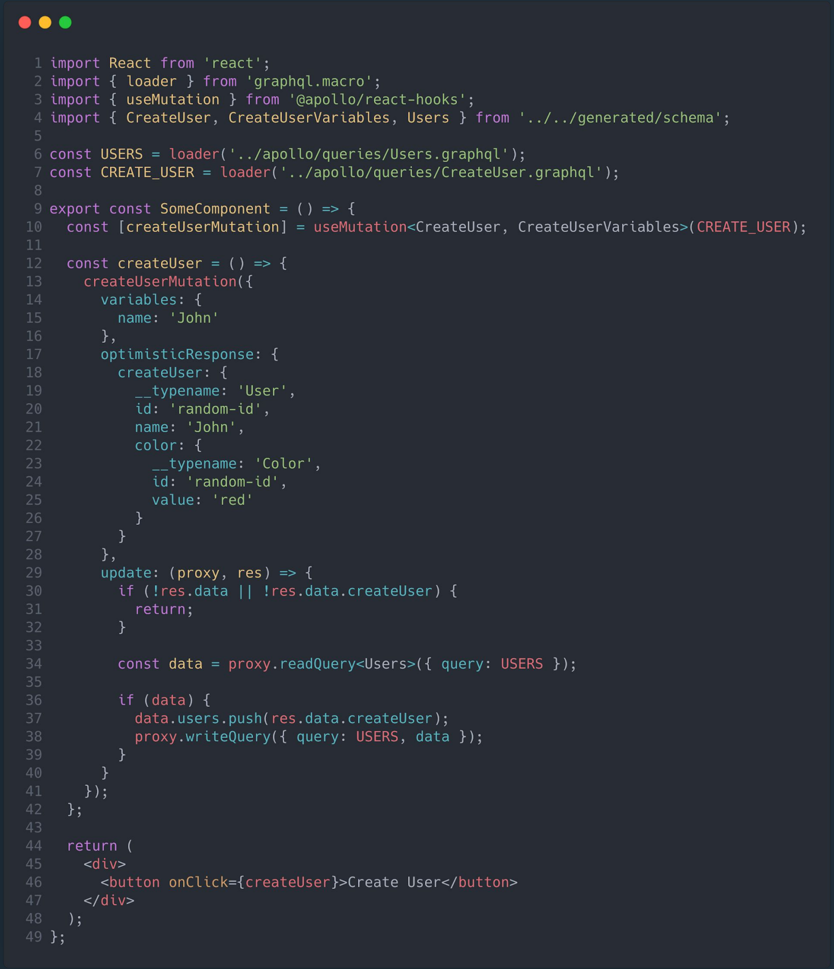
```jsx
import React from 'react';
import { loader } from 'graphql.macro';
import { useMutation } from '@apollo/react-hooks';

const CREATE_USER = loader('../apollo/queries/CreateUser.graphql');

export const SomeComponent = () => {
  const [createUserMutation] = useMutation(CREATE_USER);

  const createUser = () => {
    createUserMutation({
      variables: {
        name: 'John'
      }
    });
  };

  return (
    <div>
      <button onClick={createUser}>Create User</button>
    </div>
  );
};
```



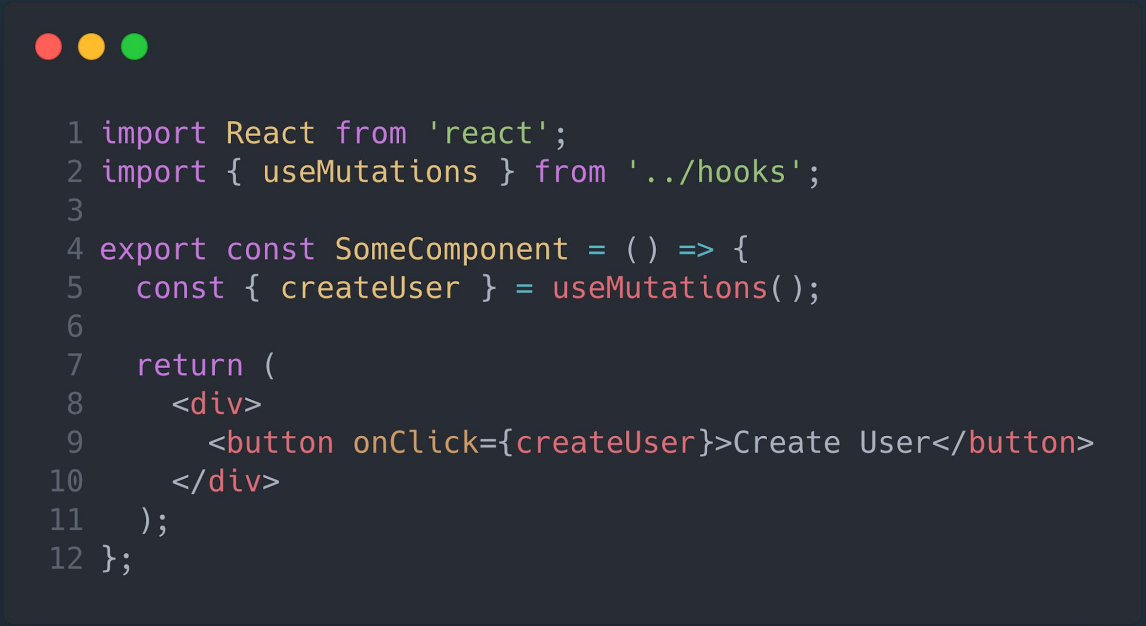
```
import React from 'react';
import { loader } from 'graphql.macro';
import { useMutation } from '@apollo/react-hooks';
import { CreateUser, CreateUserVariables } from '../../generated/schema';

const CREATE_USER = loader('../apollo/queries/CreateUser.graphql');

export const SomeComponent = () => {
  const [createUserMutation] = useMutation<CreateUser, CreateUserVariables>(CREATE_USER);

  const createUser = () => {
    createUserMutation({
      variables: {
        name: 'John'
      }
    });
  };

  return (
    <div>
      <button onClick={createUser}>Create User</button>
    </div>
  );
};
```



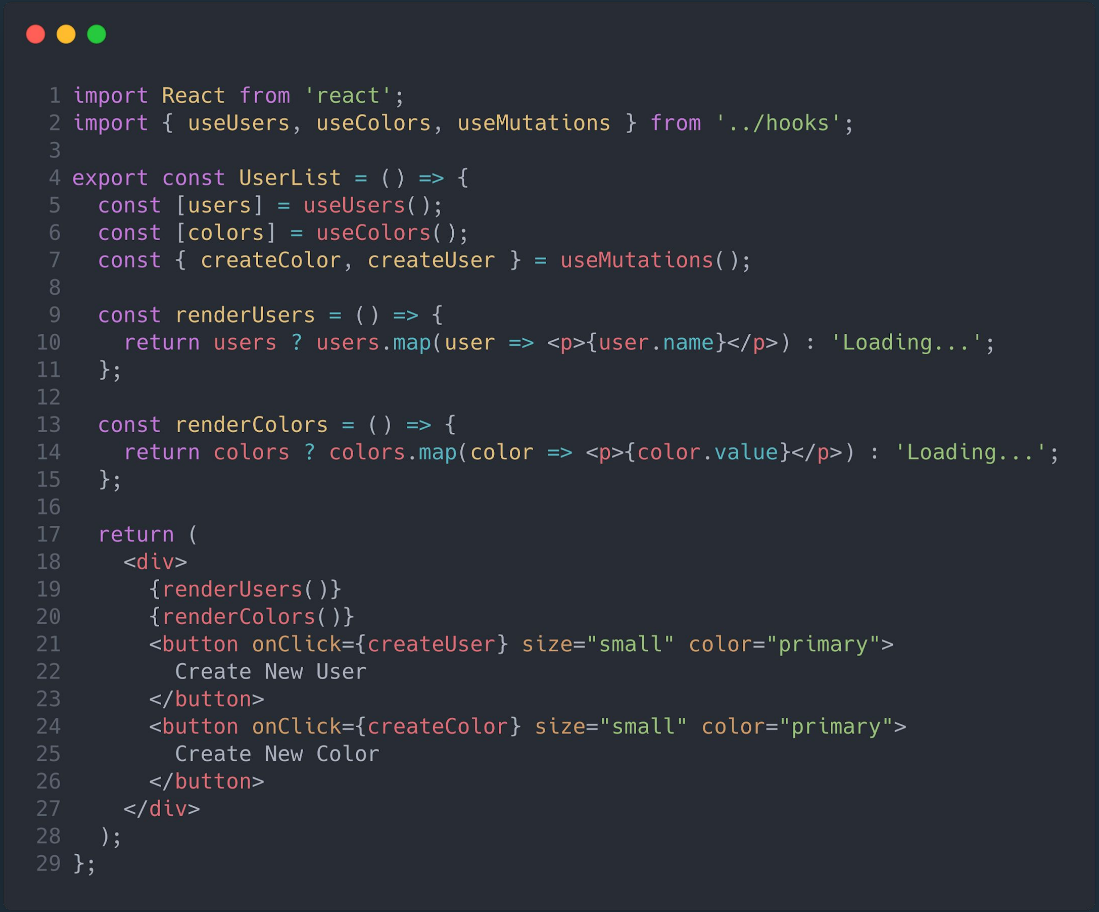
```
import React from 'react';
import { loader } from 'graphql.macro';
import { useMutation } from '@apollo/react-hooks';
import { CreateUser, CreateUserVariables } from '../../generated/schema';

const CREATE_USER = loader('../apollo/queries/CreateUser.graphql');

export const SomeComponent = () => {
  const [createUserMutation] = useMutation<CreateUser, CreateUserVariables>(CREATE_USER);

  const createUser = () => {
    createUserMutation({
      variables: {
        name: 'John'
      },
      optimisticResponse: {
        createUser: {
          __typename: 'User',
          id: 'random-id',
          name: 'John',
          color: {
            __typename: 'Color',
            id: 'random-id',
            value: 'red'
          }
        }
      }
    });
  };

  return (
    <div>
      <button onClick={createUser}>Create User</button>
    </div>
  );
};
```



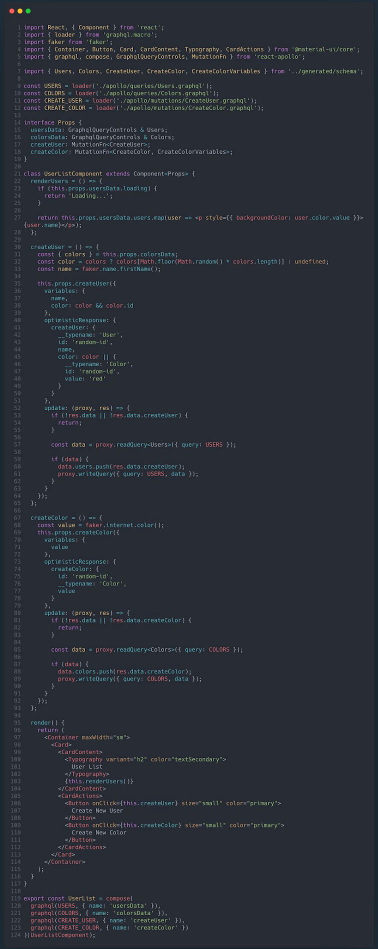
```
import React from 'react';
import { loader } from 'graphql.macro';
import { useMutation } from '@apollo/react-hooks';
import { CreateUser, CreateUserVariables, Users } from '../../generated/schema';

const USERS = loader('../apollo/queries/Users.graphql');
const CREATE_USER = loader('../apollo/queries/CreateUser.graphql');

export const SomeComponent = () => {
  const [createUserMutation] = useMutation<CreateUser, CreateUserVariables>(CREATE_USER);

  const createUser = () => {
    createUserMutation({
      variables: {
        name: 'John'
      },
      optimisticResponse: {
        createUser: {
          __typename: 'User',
          id: 'random-id',
          name: 'John',
          color: {
            __typename: 'Color',
            id: 'random-id',
            value: 'red'
          }
        }
      },
      update: (proxy, res) => {
        if (!res.data || !res.data.createUser) {
          return;
        }

        const data = proxy.readQuery<Users>({ query: USERS });

        if (data) {
          data.users.push(res.data.createUser);
          proxy.writeQuery({ query: USERS, data });
        }
      }
    });
  };

  return (
    <div>
      <button onClick={createUser}>Create User</button>
    </div>
  );
};
```



```jsx
import React from 'react';
import { useMutations } from '../hooks';

export const SomeComponent = () => {
  const { createUser } = useMutations();

  return (
    <div>
      <button onClick={createUser}>Create User</button>
    </div>
  );
};
```



```jsx
import React from 'react';
import { useUsers, useColors, useMutations } from '../hooks';

export const UserList = () => {
  const [users] = useUsers();
  const [colors] = useColors();
  const { createColor, createUser } = useMutations();

  const renderUsers = () => {
    return users ? users.map(user => <p>{user.name}</p>) : 'Loading...';
  };

  const renderColors = () => {
    return colors ? colors.map(color => <p>{color.value}</p>) : 'Loading...';
  };

  return (
    <div>
      {renderUsers()}
      {renderColors()}
      <button onClick={createUser} size="small" color="primary">
        Create New User
      </button>
      <button onClick={createColor} size="small" color="primary">
        Create New Color
      </button>
    </div>
  );
};
```



```tsx
import React, { Component } from 'react';
import { loader } from 'graphql.macro';
import faker from 'faker';
import { Container, Button, Card, CardContent, Typography, CardActions } from '@material-ui/core';
import { graphql, compose, GraphqlQueryControls, MutationFn } from 'react-apollo';

import { Users, Colors, CreateUser, CreateColor, CreateColorVariables } from '../generated/schema';

const USERS = loader('./apollo/queries/Users.graphql');
const COLORS = loader('./apollo/queries/Colors.graphql');
const CREATE_USER = loader('./apollo/mutations/CreateUser.graphql');
const CREATE_COLOR = loader('./apollo/mutations/CreateColor.graphql');

interface Props {
  usersData: GraphqlQueryControls & Users;
  colorsData: GraphqlQueryControls & Colors;
  createUser: MutationFn<CreateUser>;
  createColor: MutationFn<CreateColor, CreateColorVariables>;
}

class UserListComponent extends Component<Props> {
  renderUsers = () => {
    if (this.props.usersData.loading) {
      return 'Loading...';
    }

    return this.props.usersData.users.map(user => <p style={{ backgroundColor: user.color.value }}>{user.name}</p>);
  };

  createUser = () => {
    const { colors } = this.props.colorsData;
    const color = colors ? colors[Math.floor(Math.random() * colors.length)] : undefined;
    const name = faker.name.firstName();

    this.props.createUser({
      variables: {
        name,
        color: color && color.id
      },
      optimisticResponse: {
        createUser: {
          __typename: 'User',
          id: 'random-id',
          name,
          color: color || {
            __typename: 'Color',
            id: 'random-id',
            value: 'red'
          }
        }
      },
      update: (proxy, res) => {
        if (!res.data || !res.data.createUser) {
          return;
        }

        const data = proxy.readQuery<Users>({ query: USERS });

        if (data) {
          data.users.push(res.data.createUser);
          proxy.writeQuery({ query: USERS, data });
        }
      }
    });
  };

  createColor = () => {
    const value = faker.internet.color();
    this.props.createColor({
      variables: {
        value
      },
      optimisticResponse: {
        createColor: {
          id: 'random-id',
          __typename: 'Color',
          value
        }
      },
      update: (proxy, res) => {
        if (!res.data || !res.data.createColor) {
          return;
        }

        const data = proxy.readQuery<Colors>({ query: COLORS });

        if (data) {
          data.colors.push(res.data.createColor);
          proxy.writeQuery({ query: COLORS, data });
        }
      }
    });
  };

  render() {
    return (
      <Container maxWidth="sm">
        <Card>
          <CardContent>
            <Typography variant="h2" color="textSecondary">
              User List
            </Typography>
            {this.renderUsers()}
          </CardContent>
          <CardActions>
            <Button onClick={this.createUser} size="small" color="primary">
              Create New User
            </Button>
            <Button onClick={this.createColor} size="small" color="primary">
              Create New Color
            </Button>
          </CardActions>
        </Card>
      </Container>
    );
  }
}

export const UserList = compose(
  graphql(USERS, { name: 'usersData' }),
  graphql(COLORS, { name: 'colorsData' }),
  graphql(CREATE_USER, { name: 'createUser' }),
  graphql(CREATE_COLOR, { name: 'createColor' })
)(UserListComponent);
```

WiX Engineering

Thank You

henrikask@wix.com twitter@henry_kuzmick github.com/henrykuzmick

Q&A

henrikask@wix.com twitter@henry_kuzmick github.com/henrykuzmick