The Provider Value

In this lesson, we'll discuss how we can use the Provider value for better performance.

WE'LL COVER THE FOLLOWING

We're pretty much done with resolving the performance leaks in the application, however, there's one more thing to do.

The Issue

The effect isn't very obvious in this application but will come in handy as you face more cases in the real world such as situations where a `Provider` is nested within other components.

The `Provider` in the bank application had the following implementation:

```
...
<Provider
    value={{
       user: loggedInUser,
       handleLogin: this.handleLogin
       handleWithdrawal: this.handleWithdrawal
     }}
   >
 {this.props.children}
</Provider>
...
```

A Better Solution

The problem here is that we're passing a new object to the `value` prop every single time. A better solution will be to keep a reference to these values via the state.

For example:

For example:

```
<Provider value={this.state}>
        {this.props.children}
</Provider>
```

Doing this requires a bit of refactoring as shown below:

```
// context/UserContext.js
class UserProvider extends Component {
  constructor () {
    super()
    this.state = {
      user: null,
      handleLogin: this.handleLogin,
      handleWithdrawal: this.handleWithdrawal
    }
  }
  ...
  render () {
    return <Provider value={this.state}>
                {this.props.children}
        </Provider>
  }
}
```

Here's the complete implementation of the running project:

```
export const USER = {
  name: 'June',
  totalAmount: 2500701
}
```

Let's move on to the conclusion of this chapter in the next lesson.