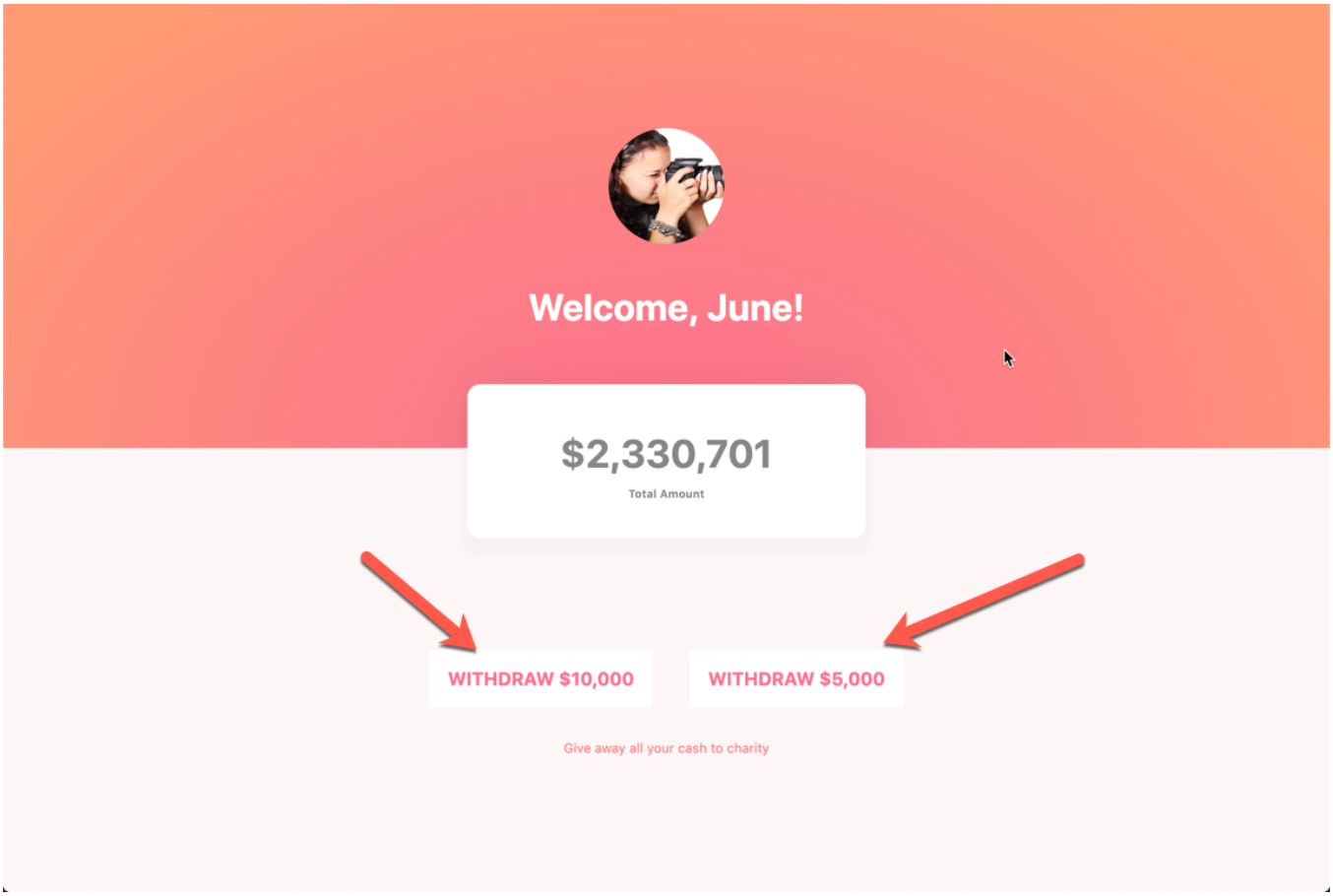
Updating Context Values

In this lesson, we'll discuss the withdrawal method of the mini-bank application.

> *What's a bank app if you can't make withdrawals, huh?*

Well, this app has some buttons. You click them and voila, a withdrawal is made.



The withdrawal buttons

Since the `totalAmount` value resides in the `loggedInUser` object, we may as well have the logic to make withdrawals in the `UserContext.js` file.

Remember, we're trying to centralize all logic related to the user object.

To do this, we'll extend the `UserProvider` in `UserContext.js` to include a `handleWithdrawal` method.

```
// UserContext.js
...
handleWithdrawal = evt => {
    const { name, totalAmount } = this.state.loggedInUser
    const withdrawalAmount = evt.target.dataset.amount

    this.setState({
      loggedInUser: {
        name,
        totalAmount: totalAmount - withdrawalAmount
      }
    })
}
```

When you click any of the buttons, we will invoke this `handleWithdrawal` method.

From the `evt` click object passed as an argument to the `handleWithdrawal` method, we then pull out the amount to be withdrawn from the `dataset` object.

```
const withdrawalAmount = evt.target.dataset.amount
```

This is possible because both buttons have a `data-amount` attribute set on them. For example:

```
<button data-amount=1000 />
```

Now that we have the `handleWithdrawal` method written out, we can expose it via the `values` prop passed to `Provider` as shown below:

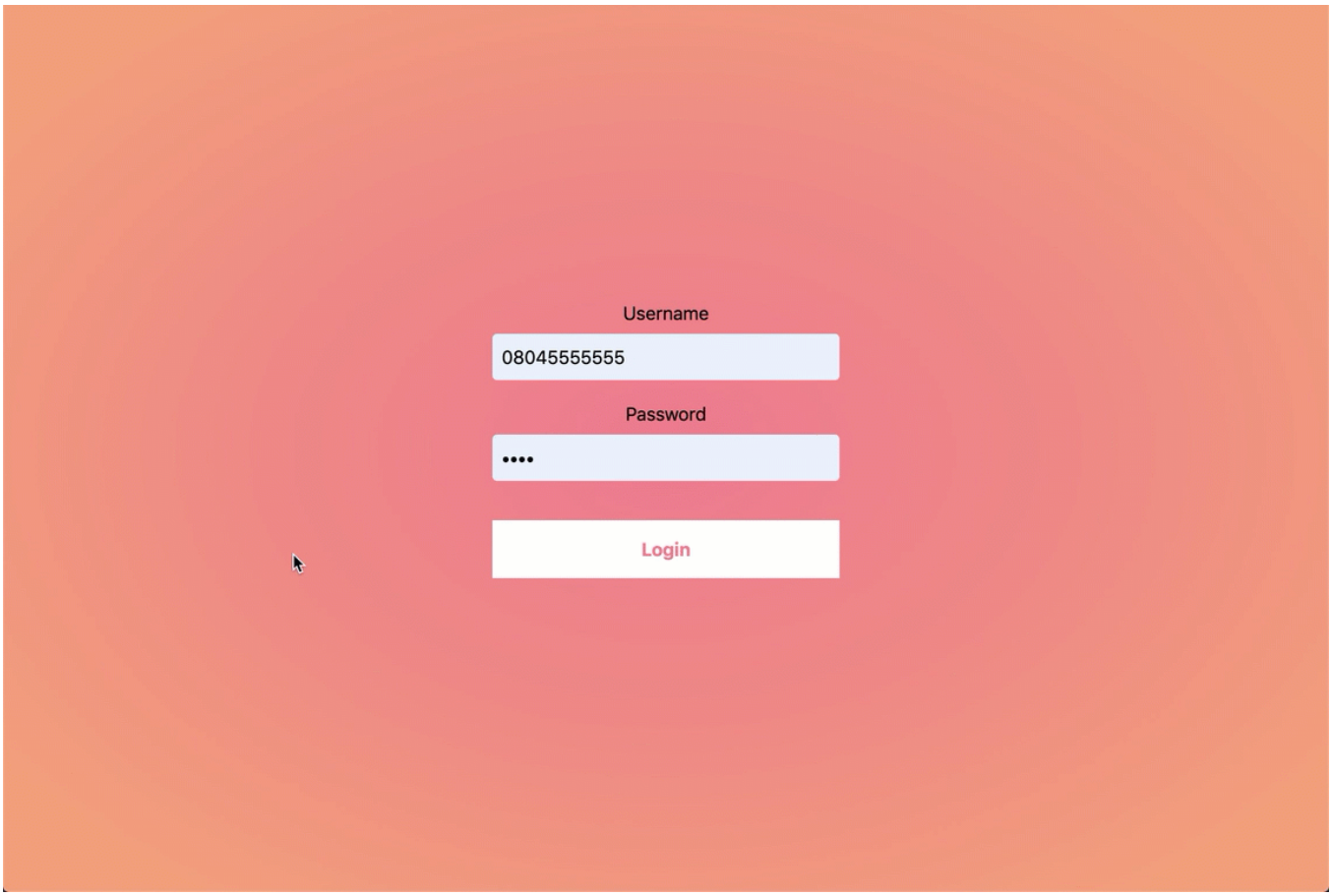
```
<Provider
    value={{
       user: loggedInUser,
       handleLogin: this.handleLogin
       handleWithdrawal: this.handleWithdrawal
     }}
   >
 {this.props.children}
</Provider>
```

Now, we're all set to consume the `handleWithdrawal` method from anywhere in the component tree.

In this case, our focus is on the `WithdrawButton` component. Go ahead and

wrap that in a `UserConsumer`, deconstruct the `handleWithdrawal` method, and make it the click handler for the buttons as shown below:

```
const WithdrawButton = ({ amount }) => {
  return (
    <UserConsumer>
      {({ handleWithdrawal }) => (
        <button
          data-amount={amount}
          onClick={handleWithdrawal}
        >
          WITHDRAW {amount}
        </button>
      )}
    </UserConsumer>
  )
}
```



On logging in, the withdrawal now works as expected.

That works!

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

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