Identifying Props being Drilled

In this lesson, we will identify and understand the props drilling in our Mini-Bank Application.

WE'LL COVER THE FOLLOWING

Root Component

The root component of the application is called `Root` and has the implementation below:

```
...
import { USER } from './api'
class Root extends Component {
  state = {
    loggedInUser: null
  }
  handleLogin = evt => {
    evt.preventDefault()
    this.setState({
      loggedInUser: USER
    })
  }
  render () {
    const { loggedInUser } = this.state

    return loggedInUser ? (
      <App loggedInUser={loggedInUser} />
    ) : (
      <Login handleLogin={this.handleLogin} />
    )
  }
}
```

Login Component

If the user is logged in, the main component `App` is rendered. If not, we show the `Login` component.

```
...
loggedInUser ? (
      <App loggedInUser={loggedInUser} />
    ) : (
   <Login handleLogin={this.handleLogin} />
)
...
```

Upon a successful login (which doesn't require any particular username and password combinations), the `state` of the `Root` application is updated with a `loggedInUser`.

```
...
handleLogin = evt => {
    ...
    this.setState({
      loggedInUser: USER
    })
  }
...
```

In the real world, this could come from an `api` call.

For this application, I've created a fake user in the `api` directory that exports the following user object.

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

Basically, the `name` and `totalAmount` of the user's bank account are retrieved and set to state when you log in.

How's the user object used in the application?

Well, consider the main component, `App`. This is the component responsible for rendering everything in the app other than the `Login` screen.

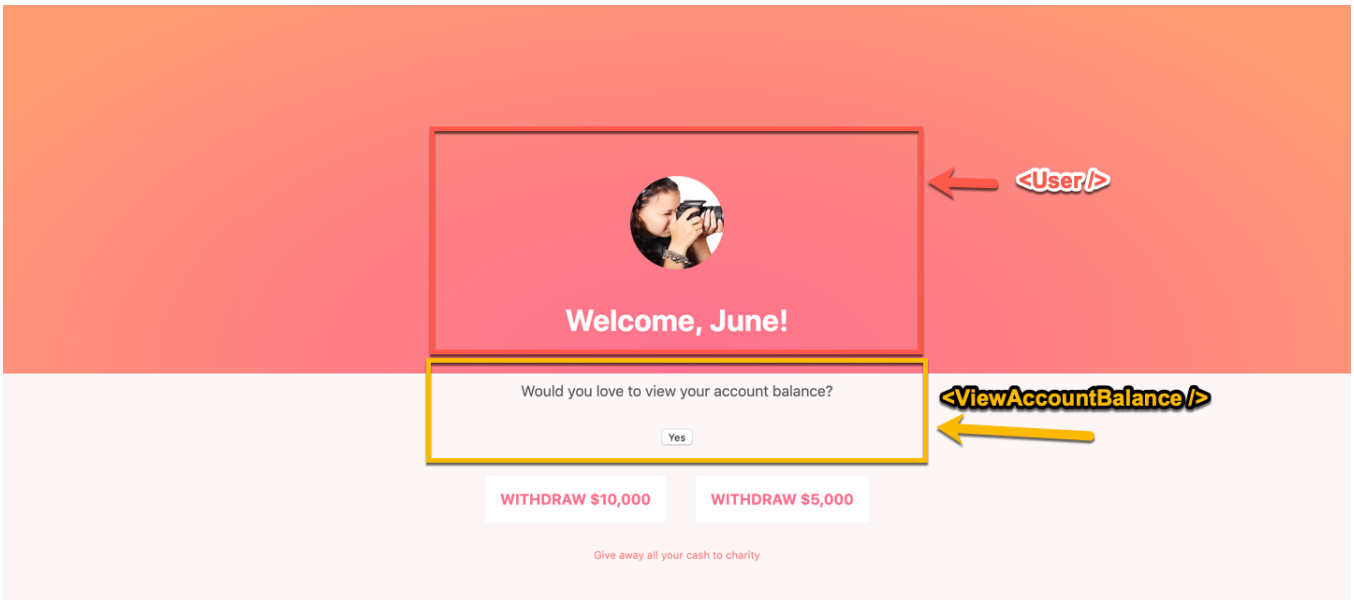
Here's the implementation:

```
class App extends Component {
  state = {
```

```
    showBalance: false
  }


  displayBalance = () => {
    this.setState({ showBalance: true })
  }
  render () {
    const { loggedInUser } = this.props
    const { showBalance } = this.state

    return (
      <div className='App'>

                <User loggedInUser={loggedInUser} profilePic={photographer} />

                <ViewAccountBalance
          showBalance={showBalance}
          loggedInUser={loggedInUser}
          displayBalance={this.displayBalance}
        />

        <section>
          <WithdrawButton amount={10000} />
          <WithdrawButton amount={5000} />
        </section>

        <Charity />

      </div>
    )
  }
}
```

It's a lot simpler than it seems. Have a second look!

The `loggedInUser` is passed as a prop from `Root` to `App`, and is also passed down to both `User` and `ViewAccountBalance` components.

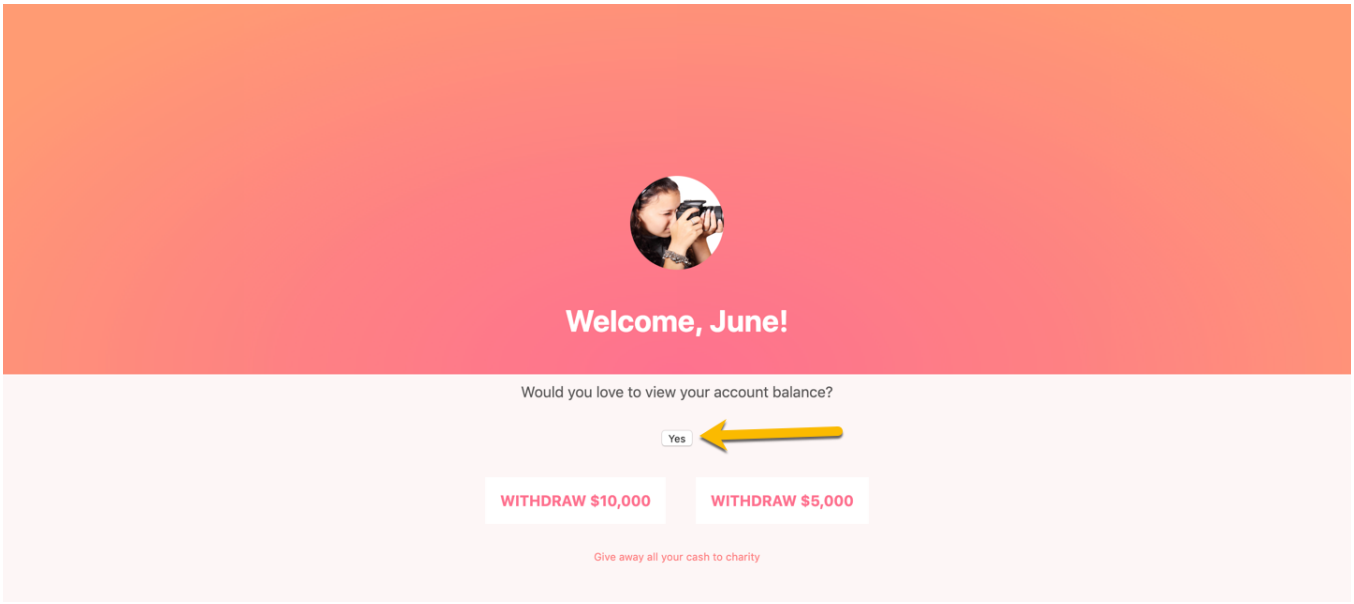


The `User` component receives the `loggedInUser` prop and passes it down to

another child component, `Greeting` which renders the text, "**Welcome, June**".

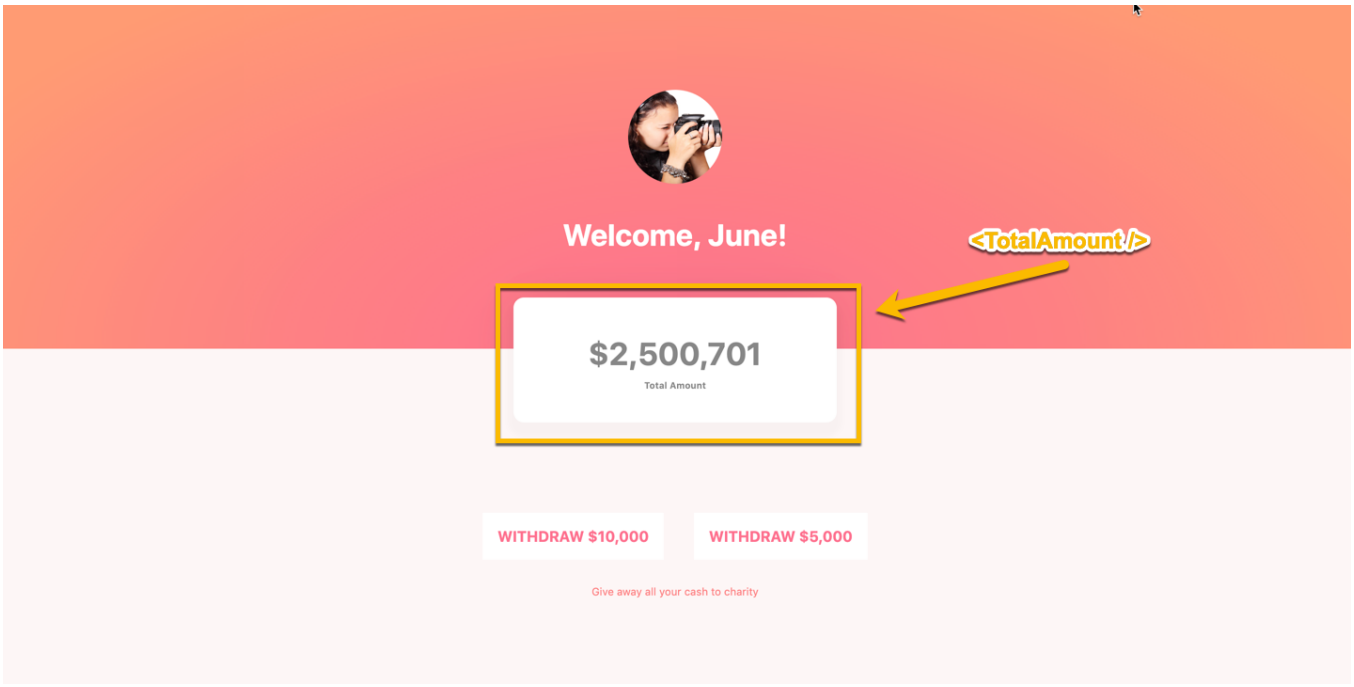
```
//User.js

const User = ({ loggedInUser, profilePic }) => {
  return (
    <div>
      <img  src={profilePic} alt='user' />
      <Greeting loggedInUser={loggedInUser} />
    </div>
  )
}
```

Also, `ViewAccountBalance` takes in a boolean prop, `showBalance`, which decides whether to show the account balance or not. This is toggled to `true` when you click the **yes** button.




```
//ViewAccountBalance.js

const ViewAccountBalance = ({ showBalance, loggedInUser, displayBalance }) => {
  return (
    <Fragment>
      {!showBalance ? (
        <div>
          <p>
            Would you love to view your account balance?
          </p>
          <button onClick={displayBalance}>
            Yes
          </button>
        </div>
      ) : (
        <TotalAmount totalAmount={loggedInUser.totalAmount} />
      )}
    </Fragment>
  )
}
```

From the code block above, do you see that `ViewAccountBalance` receives the `loggedInUser` prop only to pass it to `TotalAmount`?



`TotalAmount` receives this prop, retrieves the `totalAmount` from the user object and renders it.

I'm pretty sure you can figure out whatever else is going on in the code snippets above.

Having explained the code so far, do you see the obvious prop drilling here?

`loggedInUser` is passed down way too many times to components that don't even need to know about it.

Let's fix that with the `Context` API in the next lesson.