**Lesson 1 Review**

SEND FEEDBACK

[This](https://github.com/udacity/reactnd-UdaciFitness-complete/tree/1cc6fd9ce59e9acf6012aa5800b872ac98b5e209) is where we left off in our last lesson.

* Platform lets you know whether the app is running on iOS or Android.
* If you want a component to take up the full space of its parent, give it a flex: 1.
* We can pass multiple styles to a component by passing in an array.
* Dimensions allows you to select window's width and height in the user's device.

Let's take a look at the following piece of code:

**class** **History** **extends** **Component** {

componentDidMount () {

**const** { dispatch } = **this**.props

fetchCalendarResults()

.then((entries) => dispatch(receiveEntries(entries)))

.then(({ entries }) => {

**if** (!entries[timeToString()]) {

dispatch(addEntry({

[timeToString()]: getDailyReminderValue()

}))

}

})

}

render() {

**return** (

<View>

<Text>{JSON.stringify(this.props)}</Text>

</View>

)

}

}

**function** **mapStateToProps** (entries) {

**return** {

entries

}

}

**export** **default** connect(mapStateToProps)(History)

History is a container component. We connected it to the Redux store via the connect function. Now, this component has access to dispatch. When the component mounts, we make a call to AsyncStorage. Then, we dispatch the action receiveEntries, adding the entries received from AsyncStorage to our store. Then, if the user hasn't entered info for the current day, set the property on the redux store under the key of today to "👋 Don't forget to log your data today!".

Note how we're using mapStateToProps. The prop entries will hold everything that's in our store.

Remember that components re-render every time the state or props change. The [Redux documentation](https://redux.js.org/faq/reactredux#react-not-rerendering) outlines why a connected component may not update:

*It's important to remember that whenever you update a nested value, you must also return new copies of anything above it in your state tree. If you have state.a.b.c.d, and you want to make an update to d, you would also need to return new copies of c, b, a, and state. This state tree mutation diagram demonstrates how a change deep in a tree requires changes all the way up.*