Action Dispatch

We will be associating the setActiveUserID action we made in the last lesson with the onClick handler of our User div. This will dispatch the action on a click. We will implement this functionality in User.js.

In the last lesson we made the action creator called `setActiveUserID`, which returns the user object based on the *id* it is given.

Now we must dispatch the action.

Let's keep moving.

Take a look at the User.js component.

The first line of the return statement is a div with the class name, User:

```
<div className="User">
```

This is the right place to set up the click handler. As soon as this div is clicked, we will dispatch the action we just created.

So, here's the change:

```
<div className="User" onClick={handleUserClick.bind(null, user)}>
```

And the handleUserClick function is right here:

```
function handleUserClick({ user_id }) {
  store.dispatch(setActiveUserId(user_id));
}
```

In the code above, where has the `setActiveUserId` been imported from? The action creator!

```
import { setActiveUserId } from "../actions";
```

Below is all the User.js code you should have at this point.

Below is all the User.js code you should have at this point:

containers/User.js:

```
import React from "react";
import "./User.css";
import store from "../store";
import { setActiveUserId } from "../actions";

const User = ({ user }) => {
  const { name, profile_pic, status } = user;
  return (
    <div className="User" onClick={handleUserClick.bind(null, user)}>
      <img src={profile_pic} alt={name} className="User__pic" />
      <div className="User__details">
        <p className="User__details-name">{name}</p>
        <p className="User__details-status">{status}</p>
      </div>
    </div>
  );
};

function handleUserClick({ user_id }) {
  store.dispatch(setActiveUserId(user_id));
}
export default User;
```

To dispatch the action, I also had to import the store and call the method, `store.dispatch()`.

Also note that I have used the ES6 destructuring syntax to grab the user_id from the user argument in handleUserClick.

If you're coding along, as I recommend, click any of the user contacts and inspect the logs. You can add a console log to the `handleUserClick` like this:

```
function handleUserClick({ user_id }) {
  console.log(user_id);
  store.dispatch(setActiveUserId(user_id));
}
```

You'll find the logged user id of the user contact.

As you may have already noticed, the action is being dispatched, but nothing is changing on the screen.

The `activeUserId` isn't set in the state object. This is because right now, the reducers know nothing about the dispatched action.

We will handle the reducers in the next lesson. Fasten up!