Our completed Weather App with Redux

Let's look at Actions, Reducers and other parts of our weather app.

WE'LL COVER THE FOLLOWING

Did you try implementing the Actions left out in the last lesson. If not, try it first.

Are you done? This is what your `App` component should look like now:

```
import React from 'react';
import './App.css';
import xhr from 'xhr';
import { connect } from 'react-redux';

import Plot from './Plot';
import {
  changeLocation,
  setData,
  setDates,
  setTemps,
  setSelectedDate,
  setSelectedTemp
} from './actions';

class App extends React.Component {
  fetchData = (evt) => {
    evt.preventDefault();

    var location = encodeURIComponent(this.props.location);

    var urlPrefix = '/cors/http://api.openweathermap.org/data/2.5/forecast?q=';
    var urlSuffix = '&APPID=dbe69e56e7ee5f981d76c3e77bbb45c0&units=metric';
    var url = urlPrefix + location + urlSuffix;

    var self = this;

    xhr({
      url: url
    }, function (err, data) {

      var body = JSON.parse(data.body);
      var list = body.list;
```

```
    var dates = [];
    var temps = [];
    for (var i = 0; i < list.length; i++) {
      dates.push(list[i].dt_txt);
      temps.push(list[i].main.temp);
    }

    self.props.dispatch(setData(body));
    self.props.dispatch(setDates(dates));
    self.props.dispatch(setTemps(temps));
    self.props.dispatch(setSelectedDate(''));
    self.props.dispatch(setSelectedTemp(null));
  });
};

onPlotClick = (data) => {
  if (data.points) {
    var number = data.points[0].pointNumber;
    this.props.dispatch(setSelectedDate(this.props.dates[number]));
    this.props.dispatch(setSelectedTemp(this.props.temps[number]))
  }
};

changeLocation = (evt) => {
  this.props.dispatch(changeLocation(evt.target.value));
};

render() {
  var currentTemp = 'not loaded yet';
  if (this.props.data.list) {
    currentTemp = this.props.data.list[0].main.temp;
  }
  return (
    <div>
      <h1>Weather</h1>
      <form onSubmit={this.fetchData}>
        <label>City, Country
          <input
            placeholder={"City, Country"}
            type="text"
            value={this.props.location}
            onChange={this.changeLocation}
          />
        </label>
      </form>
      {/*
        Render the current temperature and the forecast if we have data
        otherwise return null
      */}
      {(this.props.data.list) ? (
        <div>
          {/* Render the current temperature if no specific date is selected */}
          {(this.props.selected.temp) ? (
            <p>The temperature on { this.props.selected.date } will be { this.props.selecte
          ) : (
            <p>The current temperature is { currentTemp }°C!</p>
          )}
          <h2>Forecast</h2>
          <Plot
            xData={this.props.dates}
            yData={this.props.temps}
            onPlotClick={this.onPlotClick}
```

```
              type="scatter"
            />
          </div>
        ) : null}

      </div>
    );
  }
}

// Since we want to have the entire state anyway, we can simply return it as is!
function mapStateToProps(state) {
  return state;
}

export default connect(mapStateToProps)(App);
```

Fetching Data: Going beyond xhr

We're almost done but we still have that ugly `xhr({})` call in our `fetchData` function though. This works, but as we add more and more components to our application it'll become hard to figure out where what data is fetched.

That's why the redux community has adopted `redux-thunk` as a new standard for fetching data!