Middleware

1. In preparation, take a look at your rootReducer. If you still have a console.log in there, delete it.
2. Run and test to make sure no console.log()s appear in the console.

Now, let's say that we had a requirement to allow debug logging to be turned on when needed. Middleware is perfect for that kind of thing.

3. Add a new folder called *middleware* under the store folder.
4. In your new folder, add two new files called middleware.js and logging middleware.
5. In loggingMiddleware.js, create a new function with the typical Redux middleware signature:

```
export const loggingMiddleware = ({getState, dispatch}) => next => action => {
  next(action);
  window.debugging && console.log("Just finished action", action, getState());
}
```

Remember that the *next(action)* is particularly important or no reducer will ever run.

6. In the middleware.js file, add this:

```
import { loggingMiddleware } from './loggingMiddleware';
export const middleware = [
  loggingMiddleware,
];
```

If you were to run and test right now, you'd still see nothing because middleware has to be processed and made part of the store in createStore before it will run. Let's export it from this file and import it into store.js.

7. Edit store.js. import middleware from middleware.js at the top.
8. Add a line to process the middleware into a store enhancer:

```
const storeEnhancer = applyMiddleware(...middleware);
```

   (Hint: don't forget to import applyMiddleware from Redux).
9. Then add that store enhancer as the third argument passed in to createStore:

```
export const store = createStore(reducer, initialState, storeEnhancer);
```

10. This should do the trick. Notice that we're only logging if the global object (window) has a flag called "debugging" set to true. Test that out by running again. Notice there are no logs in the console yet.
11. In the console type this:

```
debugging=true;
```

No need to stop and restart or anything. No need to change any code. Just type into the browser console.

12. Then click on a movie name or two. You'll see dispatches are being logged now. They'll continue being logged until you set debugging=false in the browser console.