Middleware Lab

In this lab we'll create and register some middleware. We're not adding any new functionality except for some logging work. No future labs depend on this one.

1. Create a new file in the store folder called middleware.js. In it, create an exported function called historyMiddleware. It should have the shape of a middleware function:

```
export const historyMiddleware= ({getState, dispatch}) => next => action => {
  // Do stuff here.
}
```

2. Put a console.warn in there to log the action and getState().
3. Make sure it has a next(action); line in it.
4. Run and test. You'll see nothing in the console because we haven't registered this function to run as middleware.
5. Edit store.js. Put this at the top:

```
import { createStore, applyMiddleware } from 'redux';
import { historyMiddleware } from './middleware';
```

6. Add this line above the createStore line:

```
const enhancers = applyMiddleware(historyMiddleware);
```

7. And add the enhancers to the createStore.

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

8. Now run and test again. You should see something logged to the console with every dispatch(). And I mean every dispatch().

Saving the state for a history

Just for fun, let's keep a history of states. We could use this for an undo functionality if we wanted to sometime in the future. Let's write every state and action to localStorage.

If you don't know how to write to localStorage, here's a review: http://bit.ly/WebStorageAPI

9. The first time we write to history, it won't exist, so create it:

```
oonst historyString = sessionStorage.history || "[]";
```

10. sessionStorage is a string, so convert it to an object/array:

```
const history = JSON.parse(historyString);
```

11. It's now an array, so you can push an object onto it. Remember that you have access to the current state via the middleware's getState() method. So something like this:

```
history.push({action, state:getState()});
```

12. Then of course put that back into sessionStorage as a string:

```
sessionStorage.history = JSON.stringify(history);
```

13. Run and test. Open your developer tools and look at what is being stored in session storage. As you make changes to your data, you should see every state get pushed to our history array.

Think about this. This could be kind of cool! If we wanted to provide an undo functionality, all we'd have to do is read back the previous state from local storage (or any previous state for multiple undoes) and write it over the current state.